January 1989 UILU-ENG-89-2204

COORDINATED SCIENCE LABORATORY
College of Engineering

LEARNING UNCERTAINTY TOLERANT PLANS THROUGH APPROXIMATION IN COMPLEX DOMAINS

Scott William Bennett

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-89-2204	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Ave. Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) Arlington, VA 22217
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Office of Naval Research 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-86-K-0309
8c. ADDRESS (City, State, and ZIP Code) Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
LEARNING UNCERTAINTY TOLERANT PLANS THROUGH APPROXIMATION IN COMPLEX DOMAINS

12. PERSONAL AUTHOR(S)
Bennett, Scott William

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	January 1989	144

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	approximation, explanation-based learning operationality, uncertainty

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Most artificial intelligence systems have difficulty functioning in complex, uncertain environments. These systems make many implicit assumptions about the world which, if inaccurate, will cause them to fail. For systems to function in these environments requires that explicit approximations be used, that approximation failures be detectable, and that the system has some method for recovering from failures. An architecture for learning approximate plans is introduced based on *explanation-based learning*. In explanation-based learning, a system uses a single observed example in conjunction with its domain knowledge to construct an explanation for how some goal in that example was achieved. This explanation can be generalized into a plan capable of functioning not only on the specific observed example but a wide variety of examples of that class. Central to explanation-based learning is a concept called *operationality* which allows a system to rate different plans. We offer a comprehensive definition for this term for use with systems functioning in real-world environments. We demonstrate how the architecture for learning with approximations can be employed to learn uncertainty-tolerant plans. An implemented system called GRASPER is described which embodies the approximation architecture and learns uncertainty-tolerant plans for grasping blocks in the robotics domain.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

LEARNING UNCERTAINTY TOLERANT PLANS THROUGH APPROXIMATION IN COMPLEX DOMAINS

BY

SCOTT WILLIAM BENNETT

B.S., Northwestern University, 1984

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1989

Urbana, Illinois

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Gerald DeJong, for innumerable fine discussions especially on approximation and operationality. He has an excellent perspective on where we have been and where we are going in explanation–based learning which ties everything together.

Our research group has always been helpful in creating a good environment for work and discussion. Special thanks go to Melinda Gervasio for many discussions on the accuracy/economy tradeoff with respect to object approximations. Thanks are also due to Ray Mooney for his work on EGGS, the generalization algorithm employed by the system, as well as early discussions on the implementation of GRASPER. Steve Chien was also helpful in the comments he offered at various stages in the work.

I thank my officemate, Larry Holder, for his numerous responses to a barrage of seemingly off–the–wall questions as well as for helping to give me a better perspective and understanding of similarity/difference–based learning techniques. Bob Reinke was also very helpful in suggesting a simplified operationality example which greatly helped in illustrating the definition.

I would also like to thank my parents who provided much love, support, and encouragement throughout my work for the degree.

This work was done at the Coordinated Science Laboratory at the University of Illinois at Urbana–Champaign. Both the implementation (Lucid Common Lisp) and the thesis (Interleaf) were done on an IBM RT Model 125. Both software and hardware used were donated by IBM. This work was supported in part by the Office of Naval Research under grant N00014–86–K–0309. The author was also supported for one semester by a Cognitive Science/AI Fellowship.

TABLE OF CONTENTS

1 INTRODUCTION

The majority of artificial intelligence systems to date have been designed to function only in simulated worlds. For instance, a system might be very good at manipulating simulated boxes around a simulated workspace. Could such a system achieve similar goals with real boxes in the real world? Naturally, the answer depends on how precisely the simulation compares to the real–world situation. With a sufficiently complex model of real–world behavior, the system could achieve its goals with high likelihood. Invariably, the higher the model's complexity, the less efficient it is to reason with. Furthermore, no matter how complex the model is, there is some chance that the system will fail. A model may be very complex and consider many factors including the potential for an excessive force causing a box to be crushed or for the weight of a box to be nonuniformly distributed. Nevertheless, even this complex model may still neglect a tear in the side of a box. In lifting the box, a gantry arm may catch on the tear and cause the box to be overturned. A system's sensors can provide information about disparities between the model and the real world. Any AI system which hopes to deal with the real world must be flexible enough to respond to such disparities. It is worthwhile to note that even a system's sensors can't provide perfect information about the world. A real–world system will inevitably need to handle uncertainty.

One promising technique for training systems to achieve real–world goals involves use of *explanation–based learning* (EBL), a method for learning plans through observation of a single training example [DeJong86, Mitchell86]. The first step of the tech-

nique is to have the system construct an explanation, using a domain theory, for how the observed example achieves some particular goal. The explanation is then generalized to form a rule which can apply not only to the observed training example but to all members of a broader class of similar examples. The formulated rule allows the system's problem–solver to perform much more efficiently when it encounters future problems of this class.

Explanation–based learning has been applied to many domains including narrative understanding [Mooney88], natural deduction [O'Rorke87], physics problem solving [Shavlik88], and robotics [Segre87] to name a few. For the most part, these systems have dealt with well–structured mathematical domains or have functioned in simplified world scenarios such as block worlds.

For example, one of the problems in robotics is known as the *robot training problem*. Robot manipulators have usually been trained either with *teach pendants*, small hand–held remote controllers which allow an operator to step the robot arm through a series of motions, or through use of various robot programming languages. Teach pendants have the advantage of being easy to use without special training. However, the sequences of motions they produce are very specific and nonrobust. Robot programming, on the other hand, can be used to construct more general programs but requires skilled programmers and can take many man–hours. Furthermore, the program is only as good as the possible situations the programmer was able to anticipate when it was written. Robotics researchers have sought a better solution to this training problem.

Explanation–based learning has been successfully applied to the problem in simplified blocks worlds [Segre87]. Here, the robot operator uses a teach pendant to instruct the robot in what operations to perform. The difference is that the EBL system acts as an intelligent observer of the operator's instructions. The system's problem solver can then learn to accomplish goals, even in ways it wasn't specifically trained for. Unfortunately, current systems make many implicit assumptions about the way the world behaves that simply aren't true. Surfaces become perfect planes, object's locations are known precisely, the gripper may be disembodied and completely accurate in its movement, and so forth.

Naturally, dealing with these differences between the real world and a simplified model world is difficult. For anyone or any system to successfully operate in the real world requires a large number of approximations to be made. If we did not make approximations, we would be swamped by the complexity of performing the simplest tasks. The most important thing about making approximations is to know that you're making them. Previous learning systems that implicitly make approximations can never reason about and therefore question them. They are doomed to make the same errors repeatedly if the error was rooted in a bad approximation. Systems which are to function in the real world must:

(1) be capable of making explicit approximations as necessary for successful operation under specified constraints

(2) be capable of detecting a failure and if it were rooted in bad approximations, identifying them

(3) have a means for refining bad approximations.

One especially important capability a real–world system must have is the capability to reason about imprecise data. Approximations can be used to represent the data and methods for operating in spite of imprecise data. These methods are described as *uncertainty tolerant*. This thesis presents a general method for learning and using uncertainty tolerant plans. Our primary focus is the robotics domain, where uncertainty tolerance is a critical factor in formulating useful plans. However, the technique discussed is not limited to robotics, and is useful wherever uncertainty and thus uncertainty tolerance is important. The next chapter discusses approximations, how plans are modeled, how operationality is viewed, and presents a general architecture for approximation in explanation–based learning. The third chapter illustrates how approximations can be used to represent uncertainties about the world. The fourth chapter, gives an overview of GRASPER, a system which implements the approximate EBL architecture to learn uncertainty tolerant grasping plans in the robotics domain. Chapters follow on how GRASPER understands observations and detects and refines approximation failures. Last, related work is discussed and future directions and conclusions are given.

2 APPROXIMATION IN EXPLANATION-BASED LEARNING

Before outlining our approach for approximation in explanation-based learning, two important areas must be addressed. First, approximations must be defined and their possible uses illustrated. Second, since our view of operationality has a major impact on our approach for learning in complex real-world scenarios, a definition of operationality is presented which takes approximate plans into consideration.

2.1 Approximations

Three of the outstanding problem areas in explanation-based learning are those characteristic of real-world domains [Mitchell86]. These are

(1) an *incomplete* domain theory may be all that is available

(2) even complete domain theories may prove *intractable*

(3) the theory the system is using may be *incorrect*.

One important feature of approximations is that they are one type of assumption: they are explicit conjectures without inferential support. Approximations, therefore,

(1) can allow otherwise incomplete theories to be completed

(2) can make otherwise intractable reasoning tractable

(3) provide a framework in which imperfections in a theory can be explained as the results of bad explicit approximations

(4) allow "learning at the knowledge-level" [Dietterich86] as approximations allow unsound inferences which can result in the discovery of more operational methods for accomplishing goals.

An approximation has the following two defining features:

(1) Assumability

An approximation must make some statement about the world based not on logical proof but on conjecture.

(2) Tunability

An approximation must provide a method by which it can be tuned as the system acquires new knowledge and/or its goals change. This is the property which distinguishes approximations from simple binary assumptions. Should a simple assumption be found inconsistent with the system's knowledge, the only choice is to remove it. Approximations are more complex and must be tunable to a tighter approximation using relatively inexpensive techniques in order to account for inconsistencies in the knowledge base.

An additional desirable feature for an approximation is

(3) Measurability

An approximation should include some method by which its quality can be rated as the system acquires new knowledge and/or its goals change. This is necessary in rating which of several approximations are most badly in need of being tuned and indirectly which is most likely to have caused some recently discovered inconsistency in the system's knowledge.

2.2 Operationality

In order to illustrate in a more formal way how approximations can be used to facilitate explanation–based learning, we move to the issue of *operationality*. One of the earliest definitions of *operationalization* for machine learning was given by Jack Mostow as: "convert[ing] knowledge about a task domain into procedures useful in performing the task." [Mostow83] The term was especially useful in representing where "learning" was taking place in EBL. As work towards a unifying framework for explanation–based learning progressed around 1986, a simple binary notion of operationality was in use. Mitchell marks specific predicates as operational, a system's goal being to express the

concept in terms of those operational predicates [Mitchell86]. DeJong and Mooney point out two disadvantages of using this approach [DeJong86]. First, no method is prescribed for how operational predicates are so designated. In effect, the procedure for marking predicates as operational is nonoperational. Second, it is undesirable to mark predicates as operational regardless of arguments. It may well be possible to judge whether a predicate can be achieved with one type of argument while impossible with another. Operationality is also dependent on the current state of the system and should therefore be dynamic not static as the earlier definition proposes. Finally, the binary definition lacks any way to express a preference between two different, yet operational, ways in which the system may achieve a goal.

2.2.1 The Model

We propose a definition of operationality which overcomes these shortcomings and includes several more specific components which should have meaning for all learning systems. In order to define operationality, a set–theoretic model for viewing plans is used.

When a system uses approximate values in constructing a plan, what the system believes the plan will accomplish and what the plan will actually accomplish when carried out in the real world can be two different things. In the model, three mappings are used. Mapping 1 is carried out under the believed functional mapping and mappings 2 and 3 are under the actual functional mapping.

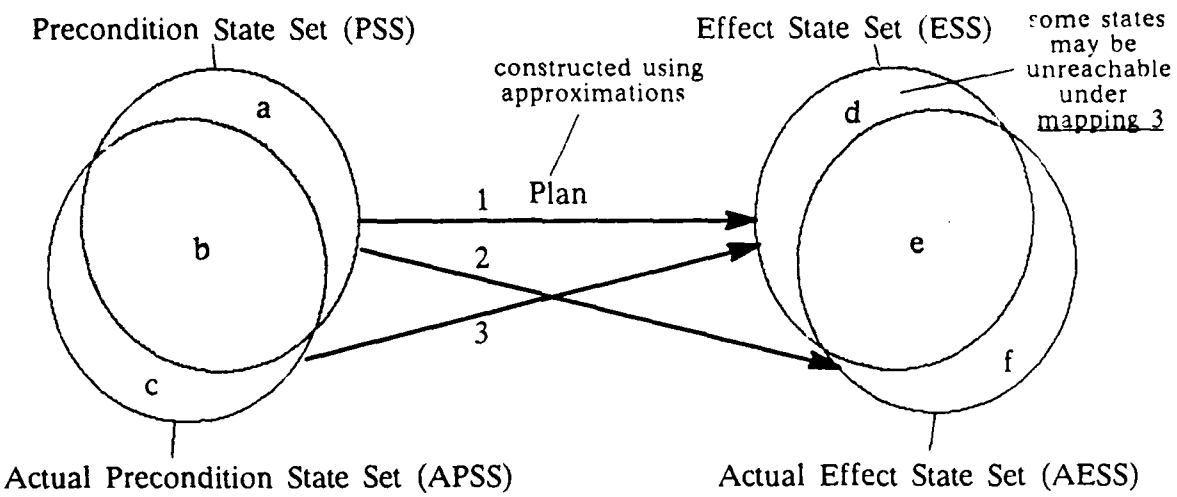


Figure 2.1. Representing the Use of a Plan

Figure 2.1 illustrates how a plan is viewed. The Precondition State Set (PSS) is the set of world states where the plan's preconditions are satisfied. The Effect State Set (ESS) is the set of states reached under mapping 1 from states in the PSS. The system treats a plan as a mapping between these two sets (mapping 1), which are illustrated by the shaded circles in the diagram. This mapping is characterized as

(1) *functional* in that it maps an x∈PSS to one and only one y∈ESS

(2) *noninjective* (not *one–to–one*) in that many states in the PSS may map to the same state in the ESS

(3) *surjective* (*onto*) in that all states in the ESS are achievable under mapping 1 from states originating in the PSS.

The shaded circles represent the system's belief about the plan. Since, the preconditions and effects to the plan were constructed using approximations, the actual mapping may behave differently when carried out in the real world. Two additional sets are introduced for this purpose. The first is the Actual Precondition State Set (APSS) and is the

set of states which the plan may be carried out from in the real world which will result in a state which is a member of the ESS. The second is the Actual Effect State Set (AESS) and is the set of states which result from application of the plan in the real world to states contained in the PSS. The plan's real world mapping between the PSS and AESS (mapping 2) is characterized as functional, noninjective, and surjective. The plan's real world mapping between the APSS and ESS (mapping 3) is characterized as functional, noninjective, and potentially *nonsurjective* in that some states in the ESS may not be achievable as a result of using approximations.

Figure 2.1 also illustrates 6 regions lettered *a* through *f*. With respect to real–world mappings 2 and 3: states in region *b* (PSS∩APSS) map to states in region *e* (ESS∩AESS), states in region *a* (PSS–APSS) map to states in region *f* (AESS–ESS), and states in region *d* (ESS–AESS), if achievable, can be mapped from states in region *c* (APSS–PSS).

The system treats the plan as if it is a mapping between the PSS and ESS as described earlier. The plan is believed to be useful when the system's current goal has a nonempty intersection with the plan's ESS (regions d2 and e2 in Figure 2.2). Note that the goal may also intersect with region f2. However, the system incorrectly believes that states in region f2 aren't reachable from the PSS. There are 8 possible cases that may arise in determining plan preconditions and subsequently applying the plan to achieve

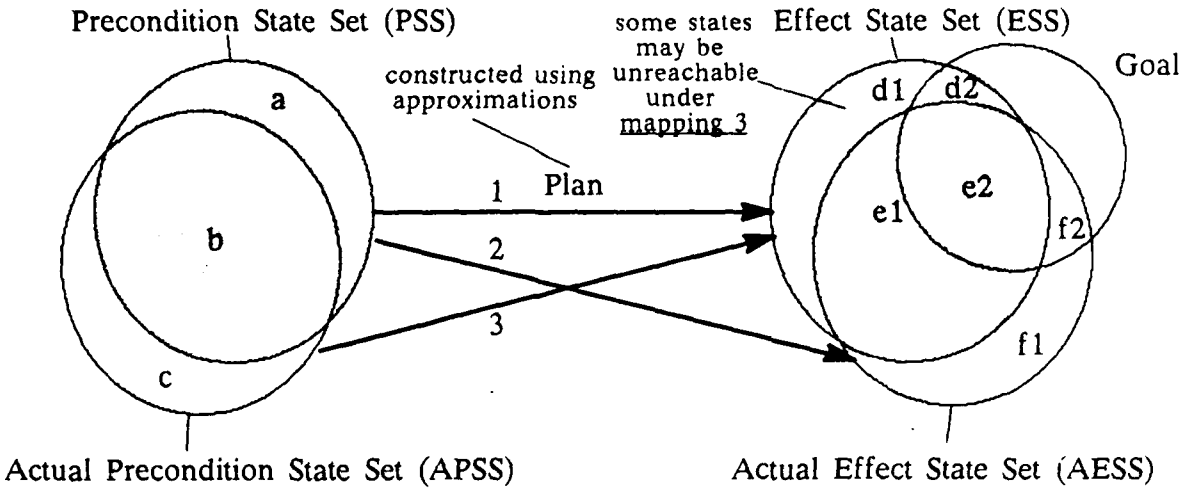
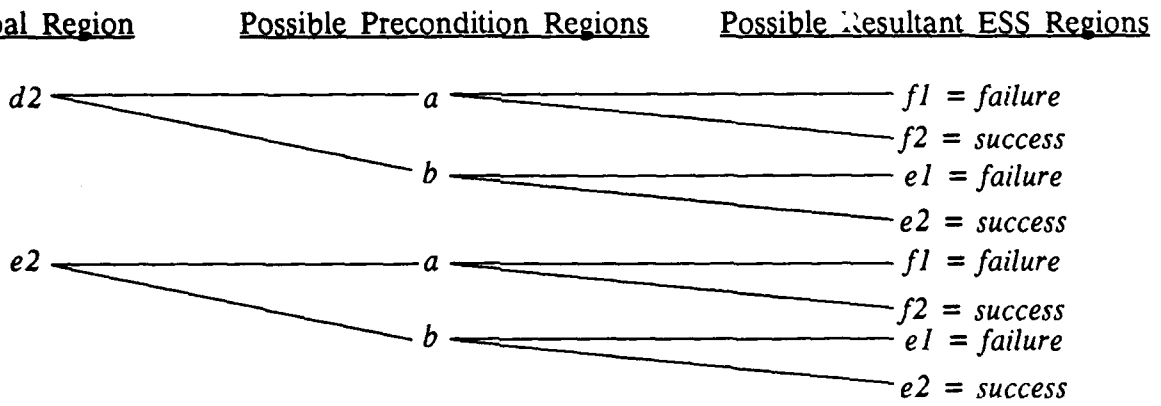


Figure 2.2. Unifying a Goal with a Plan

the goal:



Additionally, it is also possible that the system observes a plan being employed originating in a state in region *c*. If the plan succeeds in achieving a state in region *d2*, an *unexpected success* occurs. Also, some states in region *d* are not achievable as indicated by the nonsurjective nature of mapping 3.

2.2.2 An Example

Before using the model to define operationality, let us introduce a simple example which can be used to illustrate both the model just defined and the aspects of operationality. Figure 2.3 illustrates the example domain and plan. The universe of

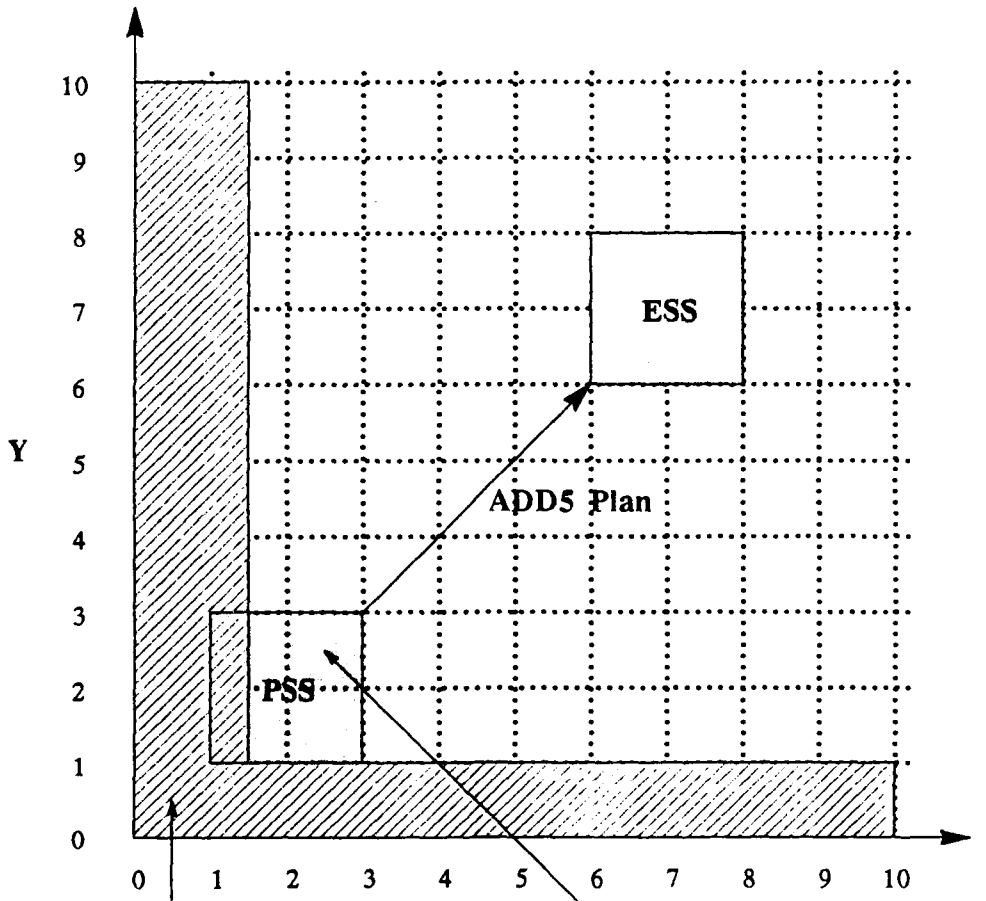


Figure 2.3. A Simple Plan with Real World Difficulties

states consists of the set of points in the 10 unit rectangle with (0,0) in the lower left and (10,10) in the upper right. The system believes that an ADD5 plan can successfully achieve states in the illustrated ESS region ($6 \leq x \leq 8 \wedge 6 \leq y \leq 8$) from those in the illustrated PSS region ($1 \leq x \leq 3 \wedge 1 \leq y \leq 3$). The figure also illustrates some difficulties

the plan encounters with respect to the real world which the system is unaware of. Whenever the plan is applied to those states contained in the shaded–L in the lower left of the diagram, the plan successfully adds 5 units to the X and Y coordinates of the point. However, with all other states the ADD5 plan succeeds only in adding 4 units to the X and Y coordinates. These simulated difficulties with the real world will give rise to a situation such as that described in the model and hence to the APSS and AESS regions illustrated in Figure 2.4. Expressions for all the state sets are given in Table 2.1. Table 2.2 demonstrates the different possible cases with respect to the goal illustrated in Figure 2.5. Figure 2.6 shows the cases as different regions within the ESS. Specific points which achieve the goal are used to show the different cases.

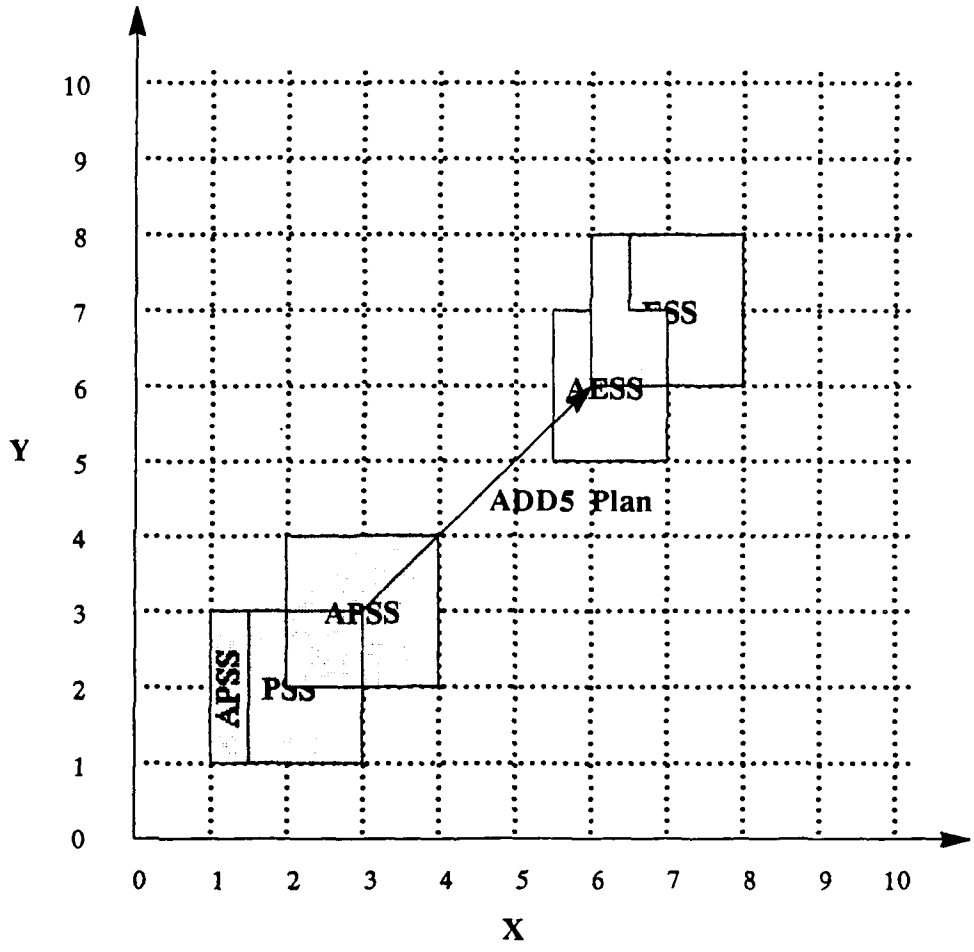


Figure 2.4. ADD5 Plan with PSS, APSS, ESS, and AESS

Table 2.1. State Set Descriptions

State Set	Region Description
Universe	$(0 \leq x \leq 10 \wedge 0 \leq y \leq 10)$
PSS	$(1 \leq x \leq 3 \wedge 1 \leq y \leq 3)$
ESS	$(6 \leq x \leq 8 \wedge 6 \leq y \leq 8)$
APSS	$(1 \leq x \leq 1.5 \wedge 1 \leq y \leq 3) \vee (2 \leq x \leq 4 \wedge 2 \leq y \leq 4)$
AESS	$(5.5 \leq x \leq 7 \wedge 5 \leq y \leq 7) \vee (6 \leq x \leq 6.5 \wedge 5 \leq y \leq 8)$
Goal	$((y \geq 5) \wedge (x \geq 6.5) \wedge (y \leq 9) \wedge (y \leq 17-x) \wedge (y \geq 1.5x-5.5)) \vee (6.25 \leq x \leq 6.5 \wedge 5 \leq y \leq 7) \vee (5.75 \leq x \leq 6.25 \wedge 5 \leq y \leq 6)$

Table 2.2. Sample States Illustrating Different Cases

Case #	Goal Type	Possible Precondition Type	Possible Resultant ESS Regions
1	*d2* (6.75,7.5)	*a* (1.75,2.5)	*f1* = Failure (5.75,6.5)
2	*d2* (7.25,6.75)	*a* (2.25,1.75)	*f2* = Success (6.25,5.75)
3	*d2* (7.2,7.5)	*b* (2.2,2.5)	*e1* = Failure (6.2,6.5)
4	*d2* (7.75,7.5)	*b* (2.75,2.5)	*e2* = Success (6.75,6.5)
5	*e2* (6.7,6.75)	*a* (1.7,1.75)	*f1* = Failure (5.7,5.75)
6	*e2* (6.8,6.75)	*a* (1.8,1.75)	*f2* = Success (5.8,5.75)
7	*e2*	*b*	*e1* = Failure (Not Possible In This Example)
8	*e2* (6.3,6.5)	*b* (1.3,1.5)	*e2* = Success (6.3,6.5)

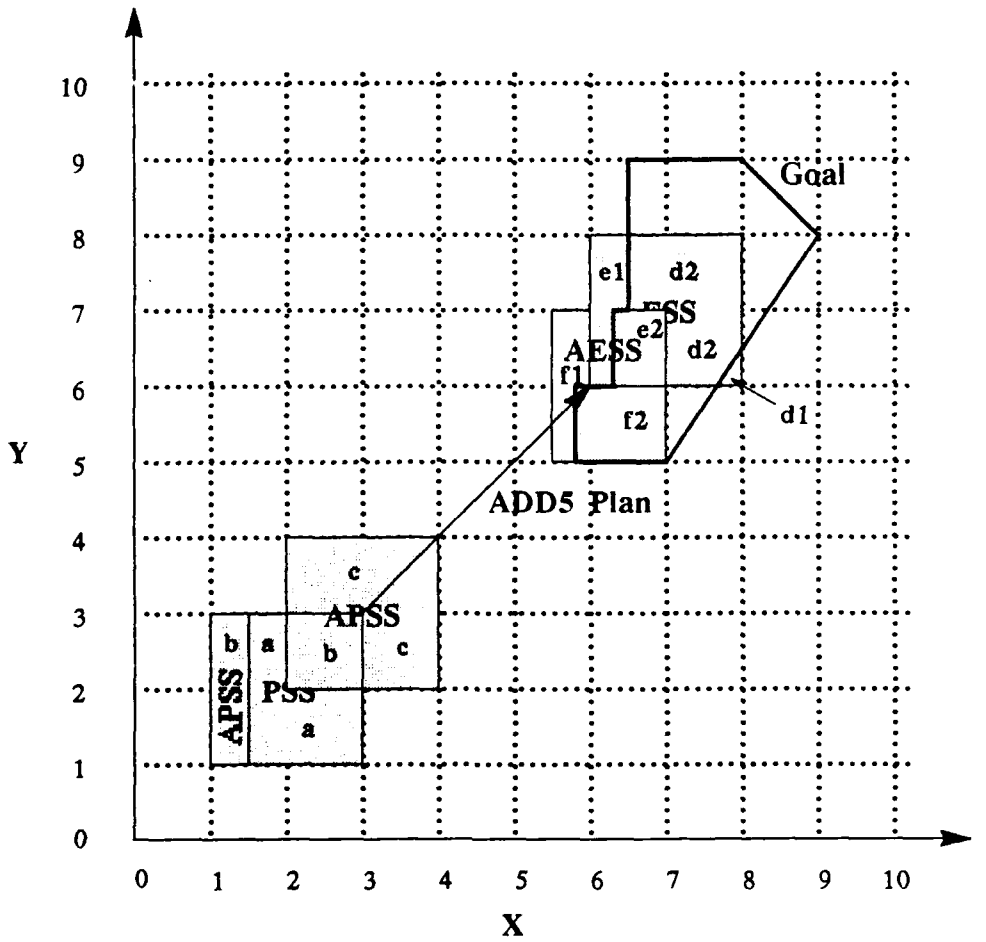


Figure 2.5. Using the Plan to Satisfy a Goal

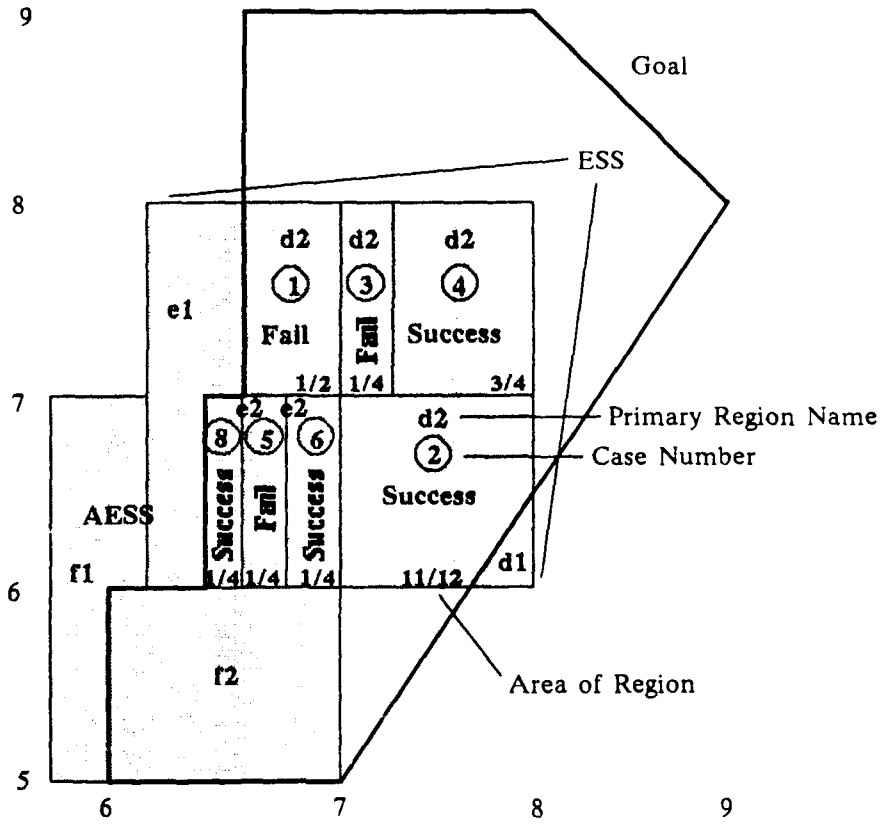


Figure 2.6. Cases Mapped to ESS Subregions

2.2.3 Aspects of Operationality

Operationality will be treated here as a continuous notion comprised of nine aspects. Each of the aspects is a scalar value normalized to $[0,1]$. Each is expressed in such a way that operationality improves as the aspect approaches 1 (holding everything else constant). In general, adjusting the plan parameters to force an improvement in one of the aspect values results in a deterioration in several of the other aspect values. The user defines a 9–ary function of the different aspects of operationality which is used to give the overall operationality determination.

After the aspects have been defined, a method is proposed for managing operationality in the context of reasoning with approximation. The definitions presented here assume world states to be equally likely as goals. The major components of operationality are as follows:

2.2.3.1 Generality Aspects

Precondition Generality (Gp)	$\frac{\lvert PSS \rvert}{\lvert universe \rvert}$

Precondition generality (Gp) refers to how broad a set of circumstances under which the plan can be applied. This is indicated by the ratio of the size of the precondition state space to the size of the universe of possible states.

It is important to note that we are defining theoretical terms here, not attempting to propose specific methods for measuring them. Ultimately, working values must be computed for these constructs. It may be tractable only to approximate these measures. That is, it is also necessary to have a way of characterizing the plan independent of the specific state to which it will be applied. Some aspects of operationality which will be discussed, such as *applicability economy,* may depend on the precise state of the world in which one is testing the applicability of the plan. This is clearly much less desirable than having adequate approximations which are less dependent on the precise situation.

Effect Generality (Ge)	$\frac{\lvert ESS \rvert}{\lvert universe \rvert}$

Effect generality (Ge) describes how large a set of world states a plan can achieve. For example, suppose plan A and plan B both have equal precondition generality. Further suppose that plan A can only achieve half as many world states as plan B. Therefore, assuming the states for both plans in the ESS are equally likely to be goals, the system would find plan B preferable. The Ge measure is necessary to express such a preference.

2.2.3.2 Economy Aspects

Applicability Economy (Ea)	$\frac{1}{1 + \text{cost}(\omega \in \text{PSS})}$

In order for a system to decide when a plan can be applied requires its preconditions to be tested. In the proposed framework, this amounts to testing whether the current world state is a member of the plan's PSS. There is always some cost associated with this operation. Plans with a low cost for this test (high *applicability economy* (Ea)) should be favored.

Effect Economy (Ee)	$\frac{1}{1 + \text{cost}(\omega \in \text{ESS})}$

In part, system performance also depends on how rapidly the current goal can be tested against available plans. This amounts to a membership test for a desired state in the ESS and is called *effect economy* (Ec).

Plan Economy (Ep)	$\frac{1}{1 + \text{cost(plan)}}$

The other major cost associated with a plan is the cost of execution. A plan may have a high Ea but still be very slow. The *plan economy* (Ep) measure is designed to reflect how inexpensively a plan's actions can be performed after it has been deemed applicable.

2.2.3.3 Real-World Aspects

In specifying formulas for the following real-world aspects the following functional notations are used:

map1(x)	with $x \in$ PSS	the resulting state in the ESS under mapping 1
map1(X)	with $X \subseteq$ PSS	$\bigcup_{\forall x \in X}$ map1(x)
revmap1(x)	with $x \in$ ESS	the set of states in the PSS which can result in x under mapping 1
revmap1(X)	with $X \subseteq$ ESS	$\bigcup_{\forall x \in X}$ revmap1(x)
map2(x)	with $x \in$ PSS	the resulting state in the AESS under mapping 2
map2(X)	with $X \subseteq$ PSS	$\bigcup_{\forall x \in X}$ map2(x)
revmap2(x)	with $x \in$ AESS	the set of states in the PSS which can result in x under mapping 2
revmap2(X)	with $X \subseteq$ AESS	$\bigcup_{\forall x \in X}$ revmap2(x)
dist(x,y)	with states x & y	a measure of the difference in certainty between an expected state x and some other state actually observed y

Probability of Success (Sp)	$\frac{\lvert \text{map2}(\text{revmap1}(\text{ESS} \cap \text{Goal})) \cap \text{Goal} \rvert}{\lvert \text{ESS} \cap \text{Goal} \rvert}$

An important factor for every system to consider is *probability of success*. This measure is based on the system's own estimation of the quality of the approximations in use. As better approximations are used, the probability of success increases. This is evident from the above definition because mapping 2 becomes closer to mapping 1 as approximations improve. How significantly probability of success is weighted depends on the degree to which the system can tolerate failure. One can envision a scenario, like picking up a beaker containing a toxic chemical, where probability of success may be treated as the most important aspect of operationality. On the other hand, picking up a small unbreakable object may be best done with a quick, easy grabbing action.

Result Accuracy (Ra)

$$\frac{|PSS|}{|PSS| + \sum_{\forall p \in PSS} dist(map1(p), map2(p))}$$

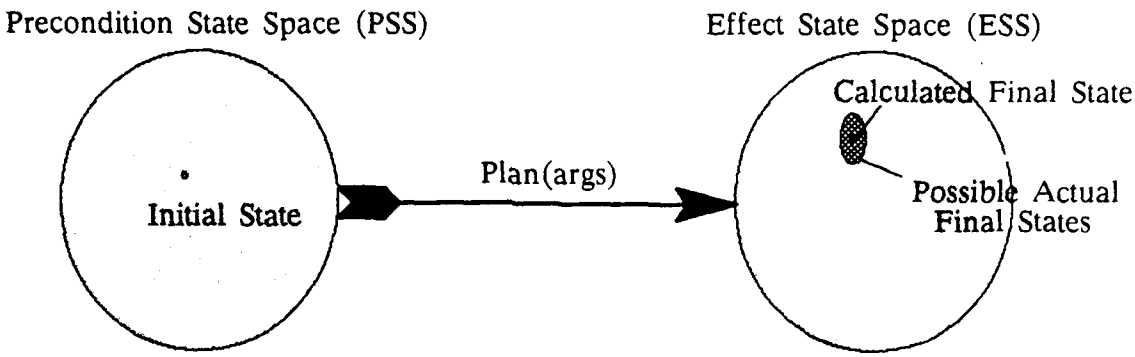


Figure 2.7. A View of Result Accuracy

If the plan's approximations are sufficiently tight, differences between what the plan accomplishes in the real world and what the system believes the plan will accomplish can be made arbitrarily small. There must exist some specification of how large this disparity is for a particular set of approximations. This measure is referred to as *result accuracy* and is depicted graphically in Figure 2.7.

Let Pg = members of the PSS which achieve the goal in the real-world
= revmap2(map2(revmap1(ESS∩ Goal)) ∩ Goal)

Uncertainty Tolerance (Ut)

$$\frac{\sum_{\forall p \in Pg} \left[\underset{\forall p2 \in (Universe - Pg)}{MIN} [dist(p,p2)] \right]}{|Pg| \underset{\forall p \in Pg}{MAX} \left[\underset{\forall p2 \in (Universe - Pg)}{MIN} dist(p,p2) \right]}$$

Uncertainty tolerance refers to the ability of a plan to achieve a goal despite uncertainties present due to a lack of knowledge about the initial world state and/or

resulting from approximations in the plan. Achieving uncertainty tolerance doesn't imply that uncertainty is reduced or even reasoned about, merely that the goal can be achieved in spite of bad knowledge. Specifically, uncertainty tolerance is the maximum amount of initial uncertainty such that the goal can be achieved. With respect to our example, those states in the PSS from which the goal can be achieved are illustrated in Figure 2.8. The definition for Ut is an average of the maximum amount by which each state in the PSS capable of achieving the goal can deviate according to the *dist* function before it fails to achieve the goal. This is then normalized to a value between 0 and 1 by dividing by the maximum possible deviation of any of these states.

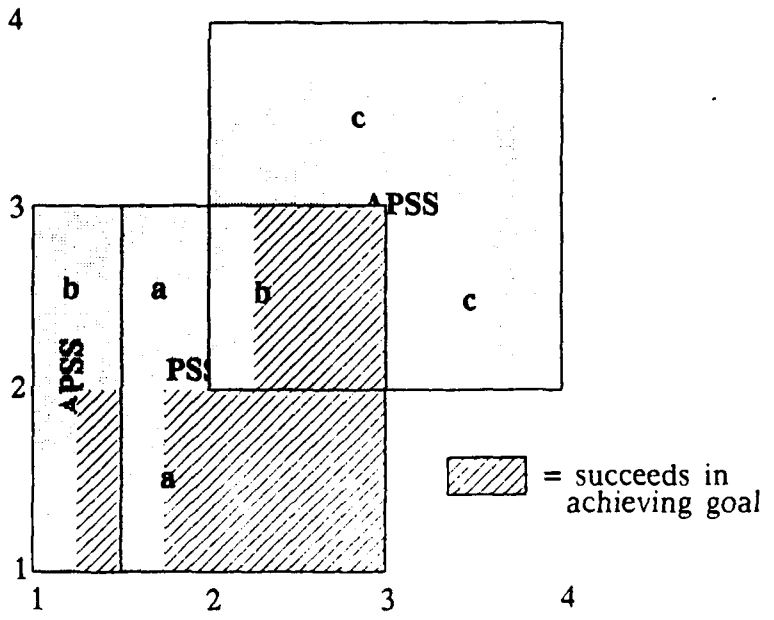


Figure 2.8. States in the PSS Which Achieve the Goal in the Real World

Uncertainty Reduction (Ur)	Ut - Ra

Uncertainty reduction takes place when a plan performs actions which serve to reduce uncertainty. It measures the "funneling" that takes place from an initially less certain situation to a later more certain situation. As with all the measures, high uncertainty reduction is preferred to the extent is doesn't sacrifice other aspects of operationality. Negative Ur is also possible and represents a plan which, when applied, leads to a less certain situation. Specifically, Ur is the difference between measures of certainty in the final situation (Ra) and those in the initial situation (Ut).

2.2.4 Measuring Operationality With Our Example

In our example, the cost of a numeric comparison is 1 and the cost of an addition is 2. The different aspects of operationality can be calculated with respect to our example as follows:

$$Gp = \frac{|PSS|}{|universe|} = \frac{4}{100} = 0.04$$

$$Ge = \frac{|ESS|}{|universe|} = \frac{4}{100} = 0.04$$

$$Ea = \frac{1}{1 + cost(\omega \in PSS)} = \frac{1}{1 + 4\ cost(numeric\_comparison)} = \frac{1}{5}$$

$$Ee = \frac{1}{1 + cost(\omega \in ESS)} = \frac{1}{1 + 4\ cost(numeric\_comparison)} = \frac{1}{5}$$

$$Ep = \frac{1}{1 + cost(plan)} = \frac{1}{1 + 2\ cost(addition)} = \frac{1}{5}$$

(areas of regions from figure 2.6)

$$Sp = \frac{\left| map2(revmap1(ESS \cap Goal)) \cap Goal \right|}{\left| ESS \cap Goal \right|} = \frac{\frac{11}{12}+\frac{3}{4}+\frac{1}{4}+\frac{1}{4}}{\frac{11}{12}+\frac{3}{4}+\frac{1}{4}+\frac{1}{4}+\frac{1}{4}+\frac{1}{4}+\frac{1}{2}} = \frac{13}{19}$$

$$Ra = \frac{\left| PSS \right|}{\left| PSS \right| + \sum_{\forall p \in PSS} dist(map1(p), map2(p))} = \frac{4}{4 + \left(\frac{1}{4} \cdot 0 + \frac{3}{4} \cdot \sqrt{2}\right)} = \frac{16}{16+3\sqrt{2}}$$

result of integrating MIN function over regions illustrated in Figure 2.8

$$Ut = \frac{\sum_{\forall p \in Pg} \underset{\forall p2 \in (Universe - Pg)}{MIN}\ dist(p,p2)}{\left| Pg \right| \underset{\substack{\forall p \in Pg \\ \forall p2 \in (Universe - Pg)}}{MAX}\ dist(p,p2)} = \frac{\square}{2\frac{1}{2} \cdot \frac{\sqrt{65}}{8}}$$

$$Ur = Ut + Ra$$

2.2.5 Variability, Granularity, and Certainty

Keller outlined several features important in characterizing operationality definitions: *variability*, *granularity*, and *certainty* [Keller87a]. Variability is either *dynamic* or *static* indicating whether the rating of a predicate as operational can change as the performance environment of the system changes. Whether our definition is dynamic depends on the specific techniques chosen to calculate the various aspects. As mentioned

earlier, various approximations may be used to make calculation of these measures easier. It is, however, desirable to have the measures linked to the environment the system is operating in. That is, they should not be linked to any specific state the plan is operating from but to the environment in general. This includes factors such as what objects are available in the world and what actions are possible. Therefore, measures such as Ea and Ee, which represent the cost of testing membership for preconditions and effects, are typically calculated based on the environment. For instance, as the number of facts, relevant or not, increases, Ea and Ee will tend to decrease as the overhead of individual predicate tests increase. Also, Ra, Ut, and Ur will be affected as they depend on the likelihood of encountering certain states, a factor which is often empirically measured and will change as the system engages in problem solving.

Granularity reflects whether operationality is defined as being *continuous* or *binary*. Our definition is a continuous one which permits making the distinction between multiple plans which are operational as to which one is the most operational.

Certainty indicates whether performance is *guaranteed* or *unguaranteed* by the operationality assessment. Although a guaranteed operationality assessment is ideal, seldom is it possible to tractably make this guarantee. One would have to anticipate all the situations in which the plan might be used and test them all to determine operationality. The components of operationality in our definition are measured using approximate means, therefore, making the assessment tractable but the result unguaranteed.

2.2.6 The Influence of Representation

One should not overlook the effect that the predicate language in which the states are expressed strongly biases aspects of operationality. For instance, increasing the fraction of states in the universe from which a plan can work (Gp) may or may not affect precondition economy (Ep). Whether it does depends on how easily the new set of states can be expressed as opposed to the former set. It also should be pointed out that in some fine grained representations the size of a state set such as the PSS or ESS may contain a very large or perhaps infinite number of states. Ideally, the representation should be course grained enough to promote tractable reasoning about operationality.

2.2.7 Dependence on the Performance Element

As Keller has observed, operationality is a function of the system's performance element [Keller88]. In our definition, we have sought to be as general as possible not tying it to any specific system's performance element. Therefore, in applying this definition for use with a specific system, the various aspects of operationality must be weighted in accordance with the performance element. If a system has a very low tolerance for failures, it may lend the greatest importance to probability of success (Sp). If it needs to deal with noisy data, it may place a premium on uncertainty tolerance (Ut) and possibly uncertainty reduction (Ur).

2.3 The Proposed Architecture

In terms of our operationality definition, approximation can be used to achieve high values for the economy and generality aspects (Ep, Ea, Ee, Gp, and Ge). The remainder of the operationality definition illustrates the price that may be paid for such improvements. Namely, the plan may not be as likely to succeed (Sp), may not achieve as predictable a result (Ra), and may increase or be unable to handle uncertainty resulting from the approximations (Ur and Ut).

An approximation could be chosen which results in an optimal plan according to the operationality definition. However, our operationality measure is *unguaranteed* as discussed earlier. Therefore, at best, the operationality definition can only provide a good approximate plan. There is simply no tractable way to anticipate all the ways in which a plan may fail. An empirical component is necessary. The plan must therefore be carried out and some recovery mechanism employed to remedy encountered problems if the system chooses to remedy them. This involves tuning the approximation(s) and proposing a replacement plan employing the new revised approximation(s). It may be the case that several methods may remedy the failure situation. In this case, the most operational of the alternatives should be chosen. In short, the set of situations in which the plan is carried out can impose constraints on the approximations. This, in turn, affects the operationality rating for the plan. However, operationality considerations should be used to favor which approximations get tuned and/or which method is used to

tune them. The general approximate explanation–based learning process therefore consists of the following seven primary steps:

(1) Approximate EBL

For the most part, this phase proceeds as with standard explanation–based learning. That is, an explanation is constructed from the system's background knowledge either through observation of an external problem–solving trace or from scratch without the guidance of such a trace. The explanation is then generalized and the resulting plan is added to the system's knowledge base. The difference between standard EBL and approximate EBL is that the domain knowledge includes explicit approximations. This has ramifications on the two learning techniques as follows:

(a) Learning From Internal Planning

While learning from planning, plans produced are based on approximations in the system's world model. This naturally means that they may fail if the approximations are bad, a possibility that must be dealt with.

(b) Learning From External Observation

In learning from observation, the system must explain observations through use of its approximate model of the world. This can lead to situations where the necessity of certain aspects of the observations are not supported by the system's approximate model. These unsupported aspects are eliminated in the produced approximate plan.

(2) Execution Monitoring

An important phase of the process involves monitoring execution of the produced plans. All action primitives in a plan carry with them a measurable expectation of behavior for the primitive in the real world. In execution monitoring, as primitives in the plan are executed, expectations are monitored. Expectation violations encountered constitute failures and the failure explanation phase begins.

At this point, the system must decide whether it will tolerate the failure or attempt to fix the plan. A system deciding to tolerate the failure continues to use the plan without any modification. Otherwise, the process continues with step 3.

(3) Failure Explanation

During failure explanation, the expectation violation, knowledge base, and world model are used to arrive at a set of plausibly bad approximations which could explain the failure.

(4) Plausibility Thresholding

Although many potentially bad approximations could have caused the failure, some may be significantly more likely to have caused it than others. The plausibility thresholding phase eliminates the potentially bad approximations below a certain plausibility threshold to promote more tractable consideration of the candidates in the next phase.

(5) Tuning Selection

In this step, the final decision is made as to which of the remaining candidate approximations to tune. This decision is made through an analysis of the new plans which result from tuning each of the candidates. This allows operationality decisions related to the resulting plan to affect the choice of a candidate to tune. Although this analysis is more expensive than other methods, it has the feature of keeping the system's current set of plans as operational as possible for the current set of approximations in use.

(6) Tuning

In the next phase, given an approximation to tune, the tuning is actually carried out by using the tuning method associated with the approximations in conjunction with the world model and knowledge base. Tuning an approximation affects the system's representation of the world model.

(7) Plan Installation

Once an approximation has been revised in response to an expectation failure, the plan is rejustified using the new world model after the approximation has been tuned. In order to relearn the plan the same observation sequence which led to the original rule being learned is reused.

The next chapter explores how approximations are used to represent uncertainty in the world and how this approach can give rise to error tolerant plans.

3 USING APPROXIMATIONS FOR ACHIEVING UNCERTAINTY TOLERANT PLANS

In this chapter, we focus on approximations which aid in the construction and refinement of uncertainty tolerant plans. First, we will discuss data uncertainty and how it is represented using the approximation framework outlined earlier. Next, the mechanism and effects of uncertainty tolerant plans are introduced. Finally, we discuss how uncertainty tolerance relates to other important aspects of operationality.

3.1 Representing Data Uncertainty

Approximations can come about in two ways:

1) *external approximation*

Approximate data may be all the system has available. For instance, a vision system may be incapable of providing precise data.

2) *internal approximation*

Even if the system has precise data, it may be forced to approximate that data due to intractability considerations. For example, a vision system might accurately characterize an object as polygons with very large numbers of vertices or generalized cylinders with many tiny intricate parts. It becomes intractable to reason about the complex polygons or cylinders in real time, so the representations are simplified by the learning system to approximate ones which contain less vertices and which the system can reason about quickly.

The effects of these are the same. In a real-world implementation, external approximation is forced by limited sensory accuracy and internal approximation is used

by the system where necessary to reach a tractable level of data for planning and understanding. The primary difference is that internal approximation can be controlled where external cannot. External approximations can be narrowed (to a more exact level) only through interaction with the world. Internal approximations can be narrowed through expending computational resources alone.

The uncertainty we are concerned with is that of the data the system receives. In constructing plans, the system reasons about possible uncertainty through use of internal approximations. Each approximation has explicitly noted the amount of uncertainty the system must plan for. A plan constructed based on approximations like these expects the values indicated by the various approximations to be correct. However, uncertainty tolerance and reduction will be employed as necessary in constructing the action sequence for the plan. In this way, the plan deals with the margins of uncertainty indicated in the approximations at the time of plan construction. Notice, that the plan itself performs no special reasoning about uncertainty when applied. Actions which have the effect of being uncertainty tolerant are introduced when the plan is constructed.

To illustrate more concretely how some of these concepts in uncertainty tolerance are applied, examples will be drawn from a 2D robotics world. Consider the sample position approximation illustrated in Figure 3.1. It is important to note that this representation is only used in constructing and revising plans. The system's world model used during plan selection and execution simply asserts the fact: (position square53 (3.5 1.0)). In the example, uncertainty is represented as a maximum distance that the object

```
(approximation
   :values
      target-type position
      object   square53
      position   (3.5 1.0)
      position-error   0.5  <-- explicit error representation
   :update-function #'position-approx-update
   :deviation-function #'position-deviation)
```

Figure 3.1. An Uncertainty Approximation Involving Object Position

could be from the designated Cartesian coordinates. That is, a plan using this approximation expects the object position to be (3.5 1.0). If the position of the square ever varies more than 0.5 units from (3.5 1.0), the approximation is in error and will require tuning. If the system deems it necessary, a new plan is constructed that incorporates sufficient uncertainty tolerance for the revised approximation. Approximations of this format break down if the position error is too great because planning or understanding becomes intractable. Our assumption is that the data given the system are not completely erroneous and that uncertainties tend to be small.

The uncertainty approximation described above is a specific type of approximation as defined in the approximate EBL framework and must therefore be tunable and measurable. That is, with the position approximation illustrated in Figure 3.1, the deviation function provides measurability and the update function provides tunability. The chapters which follow on the GRASPER system describe how these functions were implemented for the robotics domain.

3.2 Uncertainty Tolerant Plans

An *uncertainty tolerant plan* is one which can function despite some data uncertainty. There are two possible component methods to any uncertainty tolerant plan:

1) *avoidance*

This technique involves making allowances for uncertainty without necessarily reducing it. For example, consider the collision avoidance technique illustrated in Figure 3.2. A collision free path is sought for an object whose position

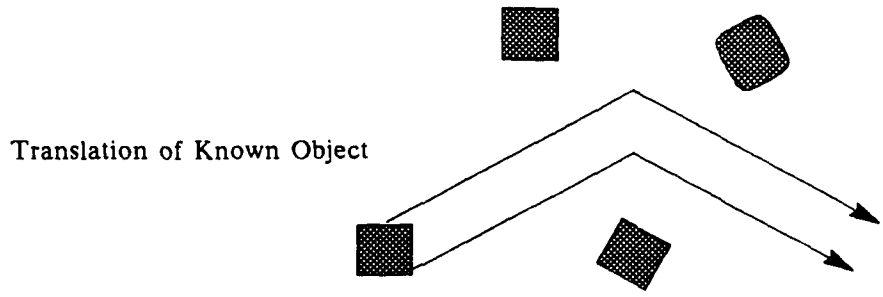


Figure 3.2. An Instance of *Avoidance*

is known through a field of objects whose positions are somewhat uncertain. The solution proposed involves a path which stays maximally far away from believed positions for objects. This constitutes one simple technique for introducing uncertainty tolerance into plans. Unfortunately, this is a rather inflexible technique — when used in isolation it can, through its conservative nature, rule out the best or only solutions.

2) *uncertainty reduction*

The technique of *uncertainty reduction* involves manipulating and sensing the world so as to reduce the amount of uncertainty. Figure 3.3 illustrates an instance of uncertainty reduction. This single translation of the gripper (which holds a long beam) made possible the more precise location of the square block. As long as the gripper and beam swept the space in which the square block was located and the gripper sensed the contact forces and torques, the block's y position is the position of the lower beam and the block is aligned with one face along

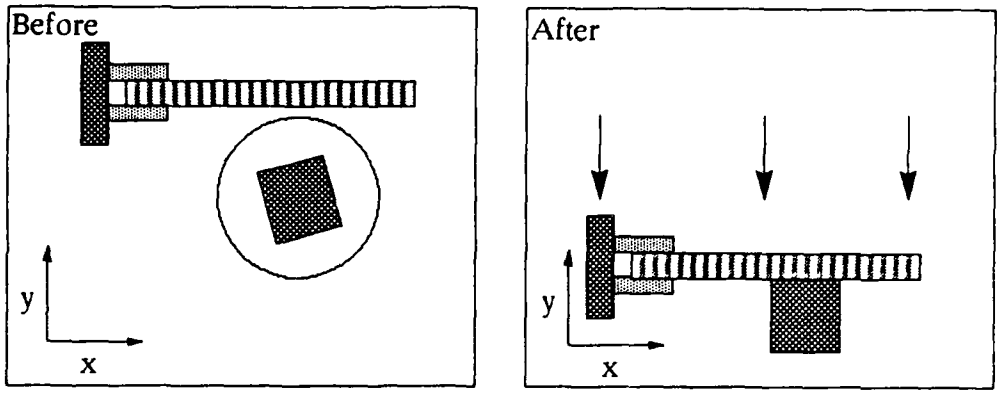


Figure 3.3. An Instance of *Uncertainty Reduction*

the beam. Overall, the maneuver has reduced the amount of uncertainty associated with the square block's location.

3.3 Recognizing and Understanding Uncertainty Tolerant Plans

A learning system must be able to recognize and understand application of both of these methods when they take place in an observed plan. The avoidance technique can be explicitly represented in the domain–specific knowledge but is not always possible and can only guarantee success under limited conditions.

The task of explaining instances of uncertainty reduction is a far more complicated one. Uncertainty reduction involves operations which achieve a state in which a set of target data is more precisely known than in the initial state. One important principle for accomplishing that is the following:

> If an external relation can be established between two internally related sets of data, the external relation can be used to reduce the uncertainty of the more uncertain of the two sets.

With this principle, the goal in uncertainty reduction is simplified to one of achieving such a relation. We seek to relate a first set of data we are more certain about to a second set we are less certain about in order to increase our knowledge of the second set.

Consider how this might be applied in a robotics domain. The uncertainty is that associated with the positions, orientations, and shapes of objects in the world. We are, however, certain of the gripper's position, orientation, and shape. One way of accomplishing uncertainty reduction is to establish a relation between the gripper and objects in the world whose associated data we would like to make more certain. To that end, an important aspect of the robotics specific domain knowledge should include a theory of object–to–object contacts and sensing. An important aspect of this theory, an instance of which is depicted in Figure 3.3, involves the concept of *alignment*. *Alignment* can be described as follows:

> If object1 is forced against object2 which is free to move without interference from other objects, object2 will tend to align itself in one of a set of possibly predictable configurations with respect to object1.

Special cases of this include the tendency for edges perpendicular to the applied force to align themselves. These could be straight, periodic, and other forms as shown in Figure 3.4. In the case of the complementary form illustrated, once the alignment has been

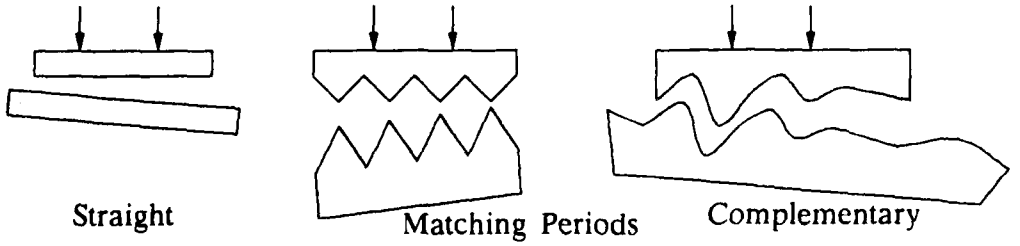


Figure 3.4. Some Instances of Alignment

accomplished, knowledge of the exact information about the top piece translates into exact knowledge about the bottom piece. For the matching periodic case, uncertainty about the bottom piece has been reduced to one of the several possible phase configurations. In the situation following straight edge alignment, position uncertainty in the x direction is still uncertain although the relative position of the two pieces in the y direction is precisely known. Figure 3.5 shows a labeled enlargement of the situation that exists after a straight–edge alignment. At plan construction time, the approximation for

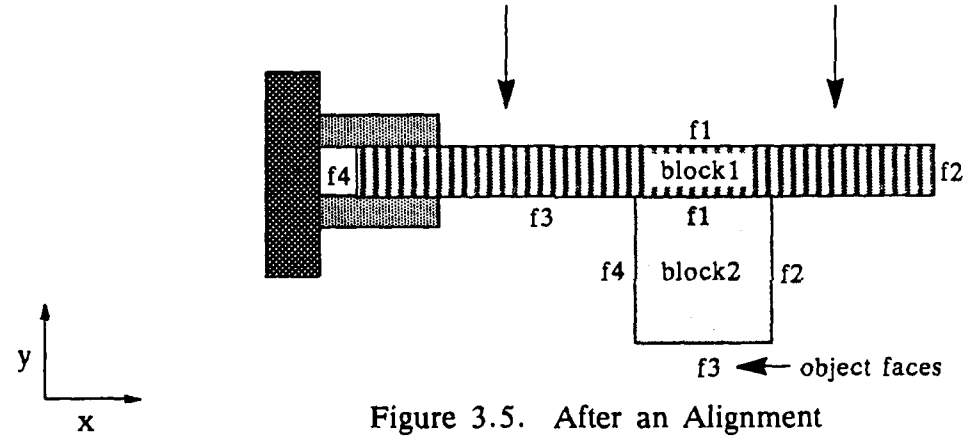


Figure 3.5. After an Alignment

block2 indicates a small possible deviation in orientation. Hence, the plan constructed specifies use of an alignment operation. When the alignment has been done, small orientation deviations are corrected. Therefore, an orientation known with certainty such as that for f3 of block1 translates into a certain knowledge of the orientation of f1 of block2.

3.4 Tradeoffs In Learning Uncertainty Tolerant Plans

One of the most important choices made about the use of uncertainty tolerance has been to reason about uncertainties only at plan creation time, not in the plans

themselves. Reasoning about uncertainties is a powerful but costly technique. In our approach, the plan is only revised if the approximations upon which it is based are questioned and the system decides it is unwilling to tolerate similar future failures of the plan. Let us consider how the goal of uncertainty tolerance interacts with the other important goals involved in learning operational plans.

Uncertainty Tolerance vs. Generality

A plan can have high generality in the sense that it is broadly applicable (Gp) and/or capable of achieving a large set of states (Ge). If an uncertainty exists for some measure, and a plan exists which has a certain degree of uncertainty tolerance for that measure, then that plan's PSS must include a range of states each of which takes on some value for that measure along a continuous spectrum. That is, the plan is still capable of achieving the goal despite slight differences in the measure. The relation between uncertainty tolerance and generality is based on the existence of such sets of states in the plan's PSS. If there is uncertainty in the environment, a plan needs uncertainty tolerance to preserve generality.

Uncertainty Tolerance vs. Economy

The most direct influence of uncertainty tolerance on economy comes in the change to economy of plan execution (Ep) due to the actions used in the plan to achieve the goal in an uncertainty tolerant manner. A plan's economy will usually decrease as extra action is often required. This, in turn, has an indirect influence on economy of applicability (Ea) and economy of plan execution (Ep) due to changes in the PSS and ESS in reflecting greater uncertainty tolerance. Economy of our plans will be much higher than those which do explicit reason-

ing about uncertainty at plan execution time. Furthermore, it is desirable to use as little uncertainty tolerance as is required by the expected problem solving situations; otherwise, economy may be sacrificed unnecessarily.

Uncertainty Tolerance vs. Probability of Success

Sometimes a system may be utilized to perform some critical application where a plan should never be executed without a great amount of certainty that it wiil function correctly. In this situation, failures cannot be tolerated. This means the system must engage in a heavy amount of reasoning and would certainly be slow and inefficient but plans produced would have a high probability of success.

On the other hand, a system might be used in an environment where failure has little or no cost. Here, trying things out can be much more efficient than performing any nontrivial reasoning about their probability of success. Each individual failure need only help in a small way in taking the system to its ultimate goal of learning the best set of plans.

Most systems will treat probability of success somewhere between the two extremes. Failure is no great tragedy but it has a nonnegligible cost. Therefore, the system should take maximum advantage of failures that do occur to try to converge quickly to a plan which fails infrequently on the situations the system observes.

In our system, we use a notion of success probability of the last variety: failures are relatively cheap but not negligible. Uncertainty tolerance has a direct contribution to probability of success for those plans operating in real-world environments where uncertainty is present.

3.5 Managing Uncertainty Tolerance

In designing a system with uncertainty tolerance, we make the following observations based on the previous discussion of tradeoffs:

(1) unnecessary use of uncertainty tolerance hurts economy

(2) lack of uncertainty tolerance in spite of uncertain data precludes most real–world applications

Figure 3.6 shows an error distribution for some measure used in a plan. A plan will be

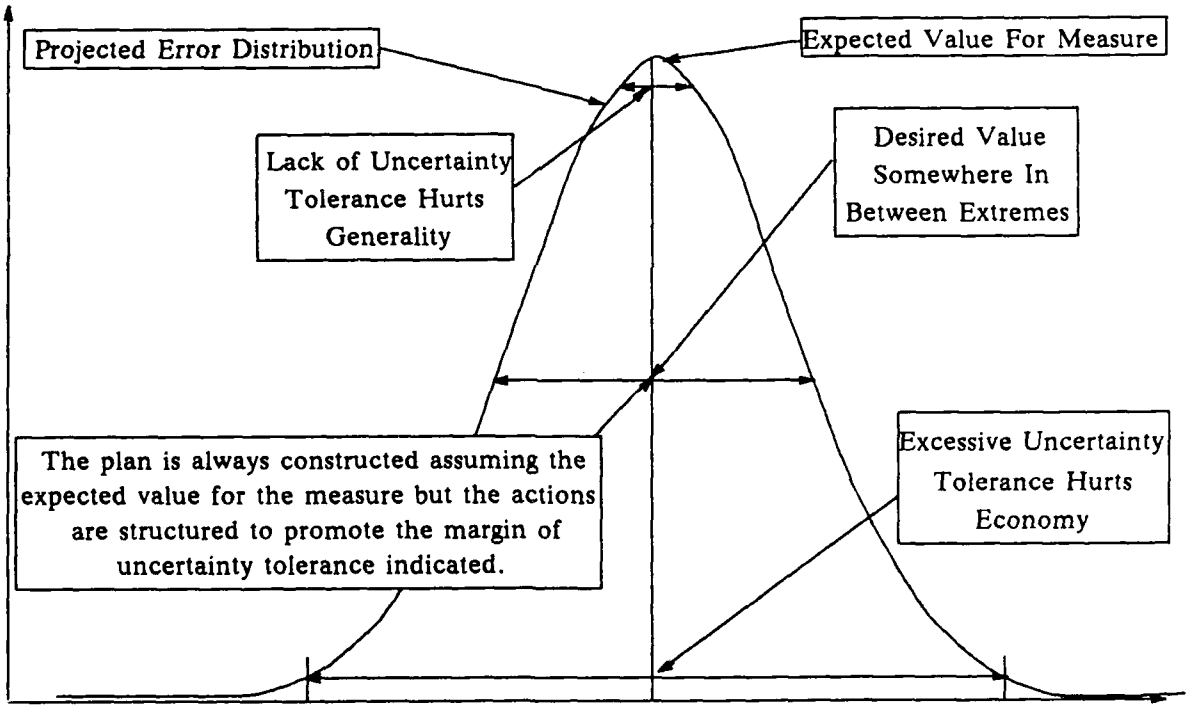


Figure 3.6. Uncertainty Tolerance, Generality, and Economy

constructed assuming the expected value for the measure. However, a decision has to be made as to how uncertainty tolerant the plan should be. On one end of the spectrum, illustrated by the horizontal double arrow at the top of the curve, a plan incorporates

little uncertainty tolerance. The generality of such a plan would be very low in an environment with uncertainty. The other extreme, illustrated by the horizontal double arrow at the bottom of the curve, incorporates a large amount of uncertainty tolerance. so much so that it even deals with uncertainties that occur very rarely. This type of plan sacrifices efficiency. What is sought is a plan somewhere between the extremes. The approximate EBL architecture outlined in the previous chapter is ideal for achieving such uncertainty tolerant plans. When failures occur, the system has a choice of redesigning the plan to incorporate another level of uncertainty tolerance or tolerating such failures in the future. This will be demonstrated by the GRASPER system which embodies this approach and is described in following chapters.

4 GRASPER SYSTEM OVERVIEW

The GRASPER system described in this chapter is intended as a first-generation EBL system with explicit approximation capability. First, the general architecture for the system will be described. Next, an example involving planning uncertainty tolerant grasping operations in robotics will be described which illustrates how the approach works.

4.1 System Architecture

Figure 4.1 illustrates how the GRASPER system is organized. The understander portion of the system observes operations being planned and carried out by an external agent. In the case of a robotics system, the understander is monitoring the command the operator gives to the robot through the teach pendant. This is similar to the idea of a "learning apprentice," but emphasis is placed on having the system function in a non-obtrusive way. The system could become a hindrance to the operator if it posed questions about the operator's intent in performing some action. Furthermore, it may be difficult for the operator to explain the intent behind every action. This problem is evident in the design of expert systems where experts can't always explain how they made a decision. The understander continually makes inferences using its knowledge base and world model as to what goals the operator is achieving and how they are being achieved. It is important to point out that where many previous systems assumed rules in the knowledge base to be guaranteed and their model to be correct, this is not neces-

sarily the case with those of GRASPER which has been designed for handling explicit approximations.

The generalizer receives explanations as to how goals have been achieved from the understander. The explanations are then generalized into efficient rules for accomplishing tasks of that class. These efficient rules are added to the system's knowledge base.

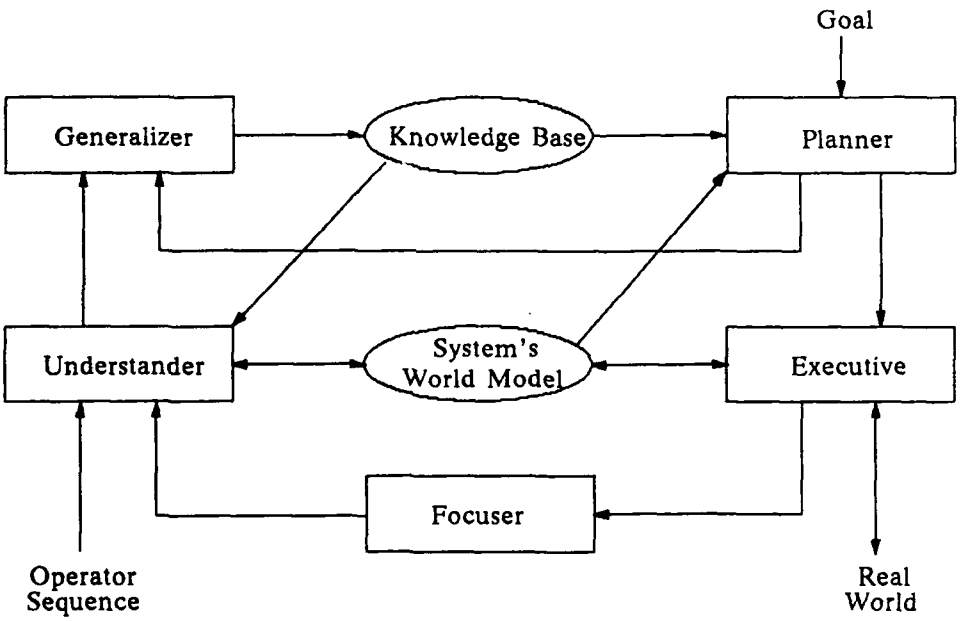


Figure 4.1. GRASPER System Architecture

GRASPER can be asked by the operator to achieve goals. The planner will attempt to construct a plan using knowledge the system has. Due to the extreme complexity of real–world domains, the planner, working with finite resources, may not be able to construct plans without the help of the understander and generalizer which com-

prise the *learning element* of the system. With them, efficient rules can be used which were already constructed through observation of external problem–solving behavior.

The executive is responsible for carrying out the plans produced by the planner in order to accomplish the desired goal. It uses the system's world model to maintain an expectation of the way the world should behave as execution progresses. Feedback is obtained from the world. For instance, when GRASPER is used in the robotics domain, various sensors are used to give position and force feedback. If the executive senses a conflict between its expectations of the world's behavior and the actual world's behavior as determined through the feedback, it recognizes a failure to have occurred. The system uses the failure to focus on finding possible fixes for it in observed behavior.

In some cases the system has already observed actions which can deal with the failure. The problem is that the system may have used approximations during understanding which obscured uncertainty tolerant aspects of the external agent's plan. It is the duty of the focusser to

(1) determine plausible candidates from among the approximations

(2) select one or more candidates for tuning

(3) decide how each candidate should be tuned

(4) update the world model in accordance with the tuned approximations.

The focusser is responsible for using information about expectation violations to focus the understander on aspects of the observed sequence that might explain how to

avoid the failure. This is accomplished through the selection and tuning of approximations that led to the failed system behavior.

Figure 4.2 gives a flow diagram for the focusser. First, the failure explainer produces a set of possible approximation failures which explain the expectation violation. This set is filtered based on a domain–specific plausibility heuristic to eliminate possible but extremely unlikely failures. The tuning selector analyzes elements of the remaining set and decides which will be tuned. The important goals in deciding what to tune are really a function of the plan which would result from the tuning. The preferred resulting plan should be judged on its operationality according to the definition proposed in Chapter 2.

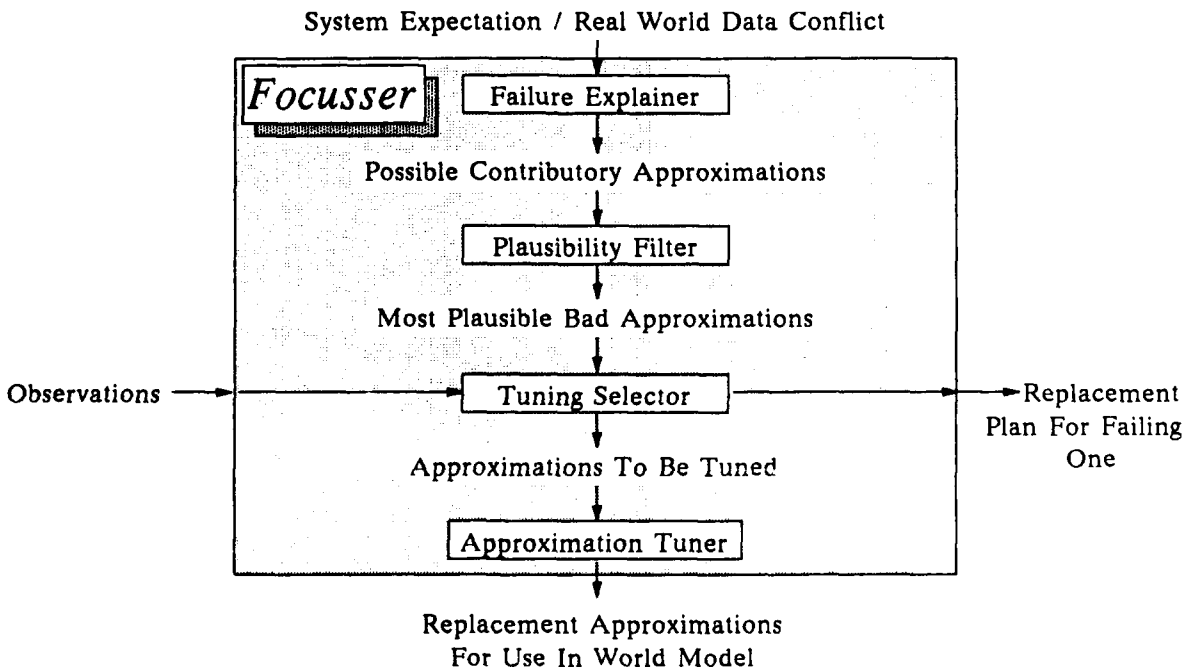


Figure 4.2. The Focusser

The focusser needs a set of observations which are ideally those associated with original construction of the rule which failed. Each candidate plan is constructed by tuning its candidate approximation and using the understander and generalizer on the observations. Once the selector completes its plan analysis, the plan can be entered in the knowledge base while the tuned approximation is included in the world model.

4.2 Use of GRASPER for Learning in Robotics

In order to illustrate how this architecture is employed, we introduce an example concerning learning uncertainty tolerant grasping strategies in the robotics domain. The example takes place in a two–dimensional world with a disembodied gripper and a set of polygonal objects. Real–world complexity is introduced through uncertainty in the known positions, orientations, and shapes of the objects. It is assumed that the position, orientation, and shape of the robot are known.

4.2.1 The Gripper

The two–dimensional gripper is shown in Figure 4.3. The two fingers slide open and closed along the beam, each finger having a motion equal and opposite to the other. The *gripper reference point* is the point at which the gripper's position and orientation are measured.

4.2.2 Gripper Primitives

Gripper motion is directed using a set of four primitive actions: *apply–force*, *apply–torque*, *close–fingers*, and *open–fingers*. The format of these actions is illustrated in Figure 4.4. All motion is compliant; the primitives specify forces to be applied. Each

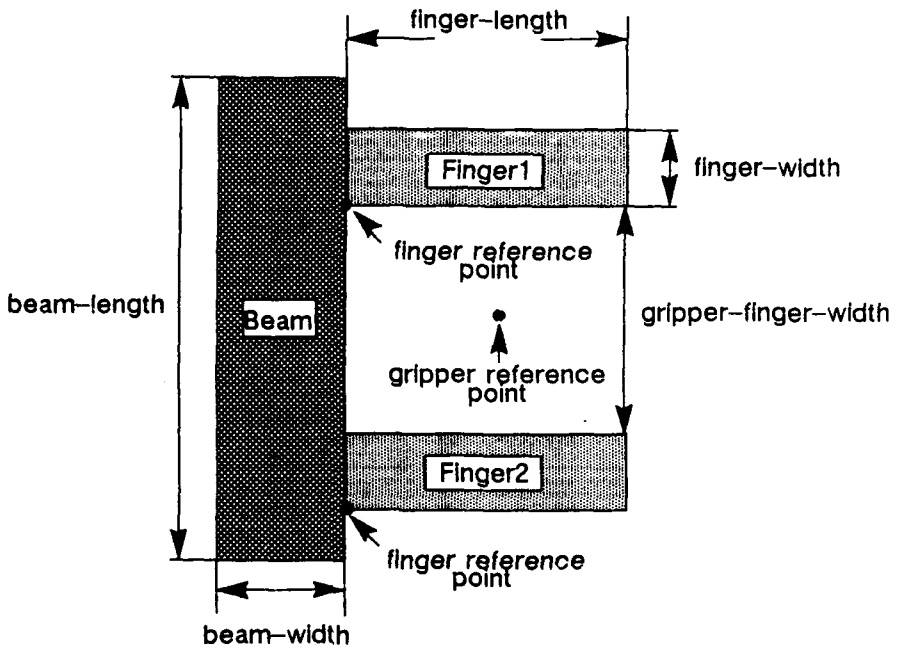


Figure 4.3. The Gripper

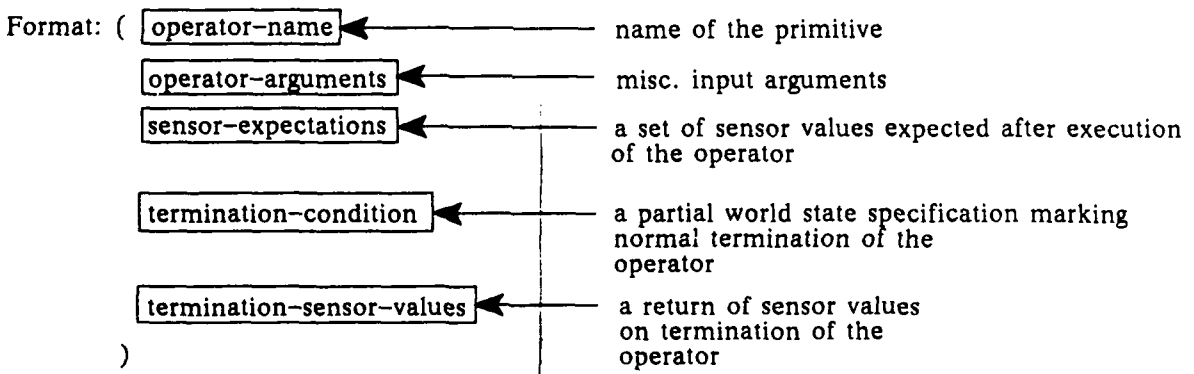


Figure 4.4. Robot Primitive Format

operator invocation also specifies expected sensor readings and a termination condition. An operation terminates if any of the sensor expectations are violated, in which case a failure occurs, or if the termination condition is met without a violation of sensor expectations.

4.2.3 Gripper Sensors

The gripper is equipped with force sensors on its periphery and with position sensors. The sensor expectations consist of a set of position and force specifications with respect to the gripper which must be satisfied at completion of the operation. The termination condition is a partial world state specification which marks completion of the operation. The termination sensor values are a dump of all the sensor values returned at the termination of the operation. Force sensors return information as illustrated in Figure 4.5.

Force representation:

(force ?finger ?contact-x ?contact-y ?magnitude ?x-component ?y-component)

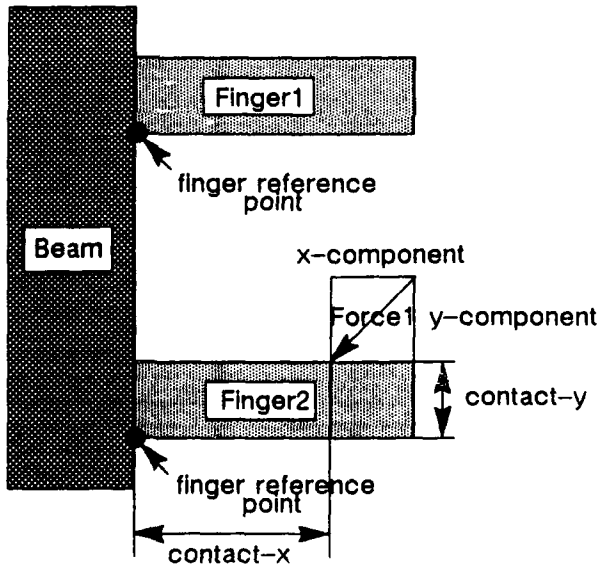


Figure 4.5. Gripper Contact Sensing

4.2.4 Object Representation

Objects in the two-dimensional world have their shapes represented by simple polygons. The polygon has a reference point about which the object's orientation and at which its position is measured. Shape, position, and orientation information about each

object in the world is contained in the system's world model. This information is treated as precise.

4.2.5 Object Approximations

At plan construction time and in explaining failures, it is necessary to reason about object shape, position, and orientation as approximations to their exact values in the real world. These approximations take a form as shown for the position approximation of Figure 4.6. The approximation has two domain–specific functions associated

```
(approximation
   :values
       target-type position
       object   square53
       position   (3.5 1.0)
       position-uncertainty 0.5   <-- explicit uncertainty representation
   :update-function #'position-approx-update
   :deviation-function #'position-deviation)
```

Figure 4.6. A Position Approximation

with it. The first, called the *deviation function*, in this case measures distance between a specified position and the position in the approximation. This is used for rating which of several approximations can be most easily extended to account for errant positions. The *update function* is used for extending the approximation to account for such a position. Figure 4.7 demonstrates the procedure for tuning the approximation to account for an errant position. In this case, one face of a polygon has been assigned an approxi-

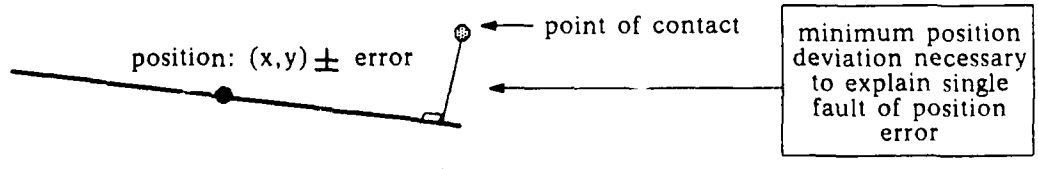


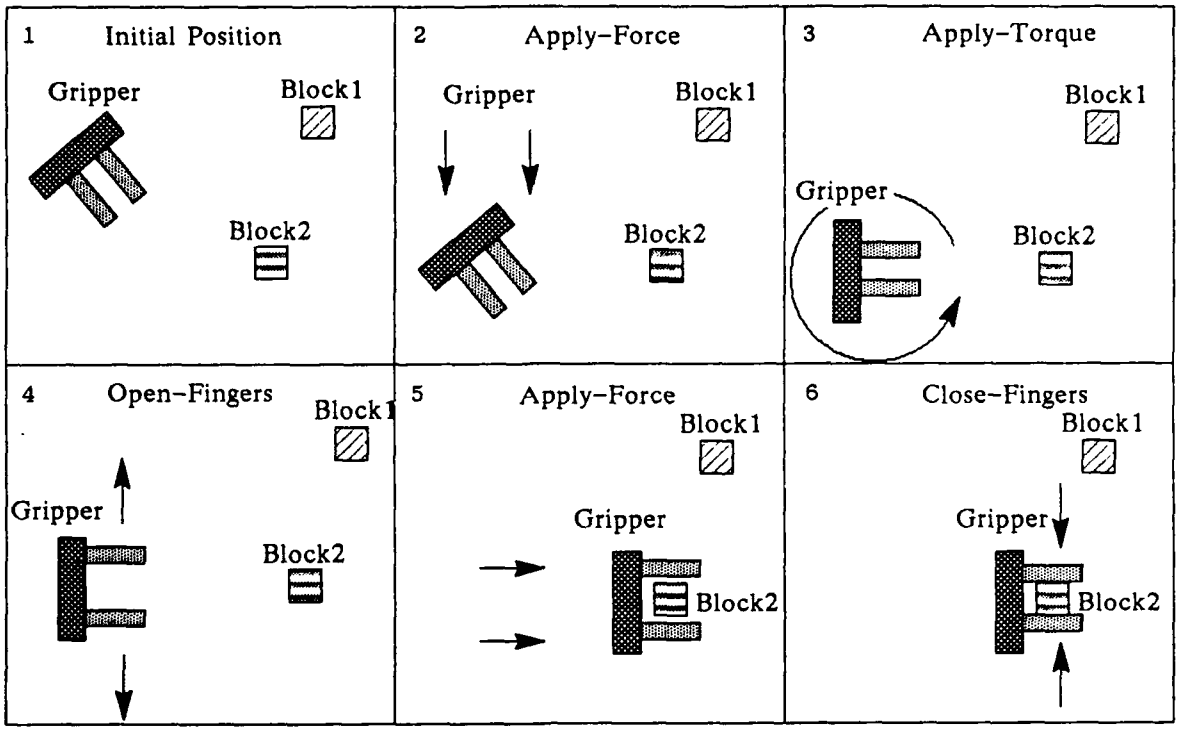
Figure 4.7. Attributing Position Error

mate position but a known contact point has forced modification of the approximation to account for it. The update function extends the value of the position uncertainty to account for it.

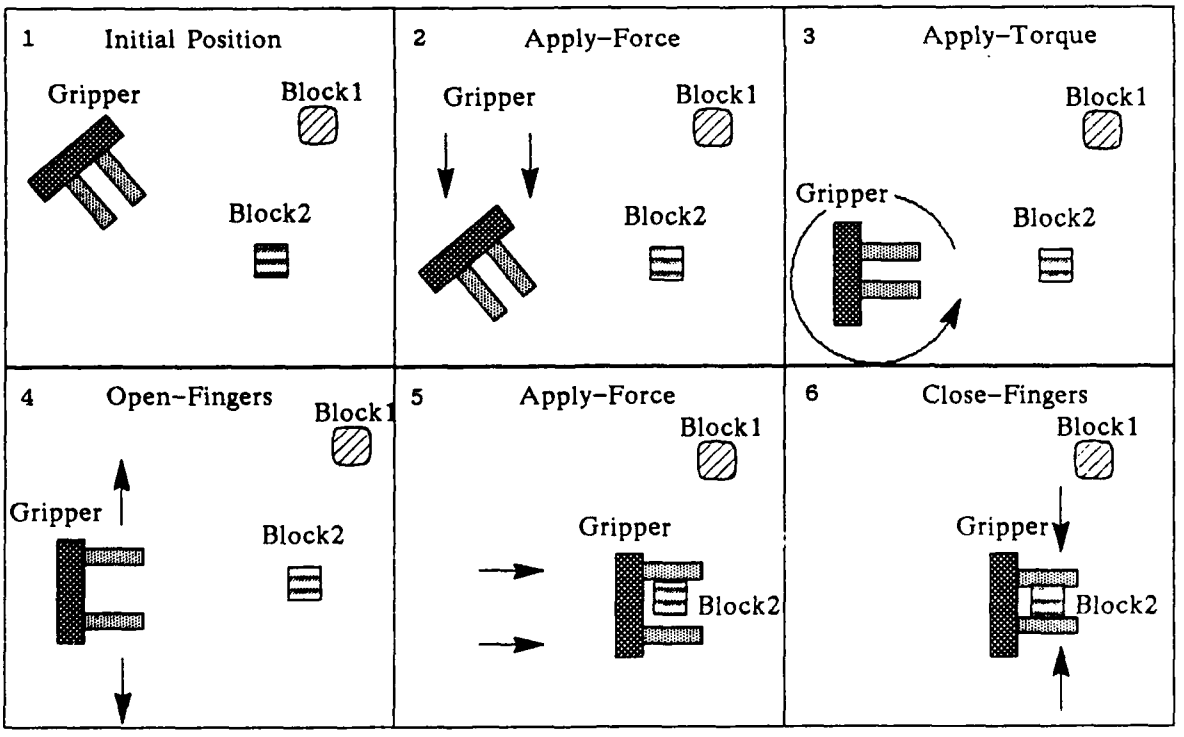
4.3 An Example

The GRASPER system starts out by observing a *grasp* operation being carried out on *block2* as shown in Figure 4.8. The figure shows the six states involved in the operation both from the approximate view of the system (4.8a) and from an ideal precise view (4.8b). The system is incapable of getting or processing information as precise as that seen in the latter view in real time. From the initial position (frame 1), an *apply–force* operator leads to a successful translation of the gripper to an approach position (frame 2). An *apply–torque* operator aligns the gripper's fingers toward the object through a successful rotation (frame 3). The *open–fingers* operation is performed to allow the gripper to be opened wide enough to surround the object (frame 4). Another *apply–force* is used to move the gripper to a position around the block (frame 5). Last, a *close–fingers* is used to bring the fingers together enough to achieve a specified gripping force on the object (frame 6).

From the initial view of the understander and the approximate view of the world, the extra–wide opening of the fingers for the approach is both inefficient and unnecessary. This view of the observed sequence finds its way into the general rule produced to achieve a grasping goal. Figure 4.9 shows how the system performs with the new rule when asked to apply it to a similar grasping situation. The plan proceeds much as the

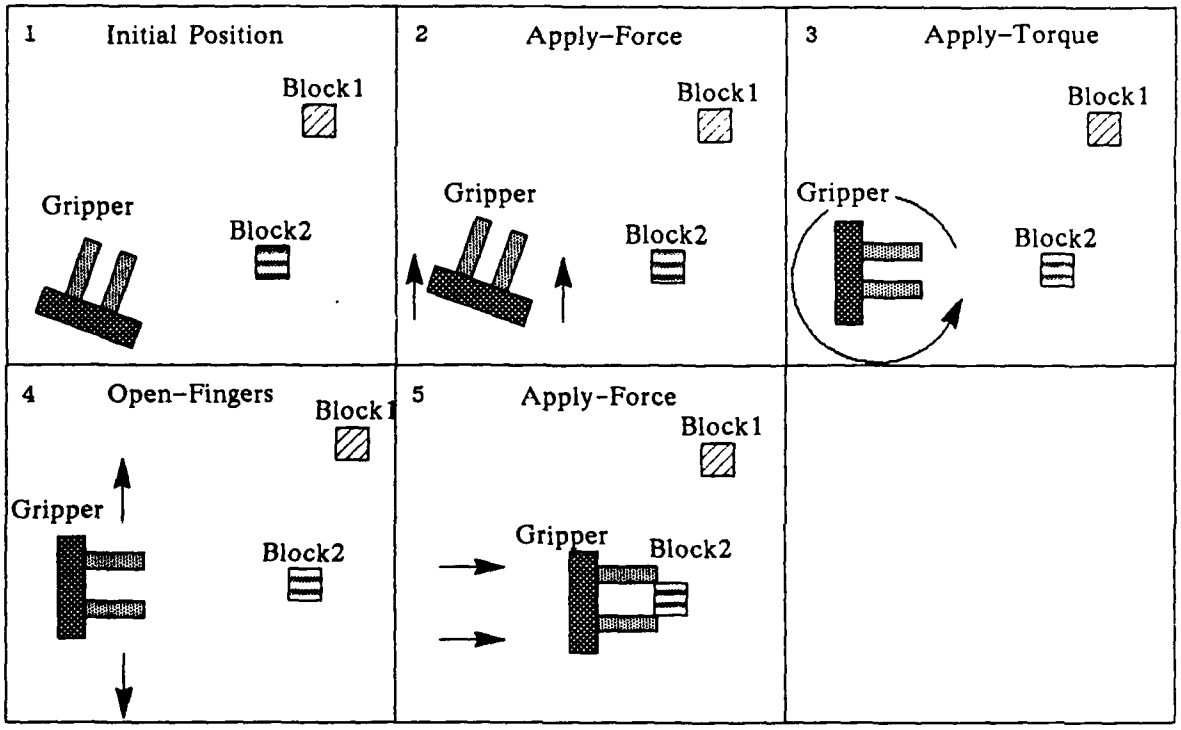


(4.8a) System's Approximate View of the Operator Sequence

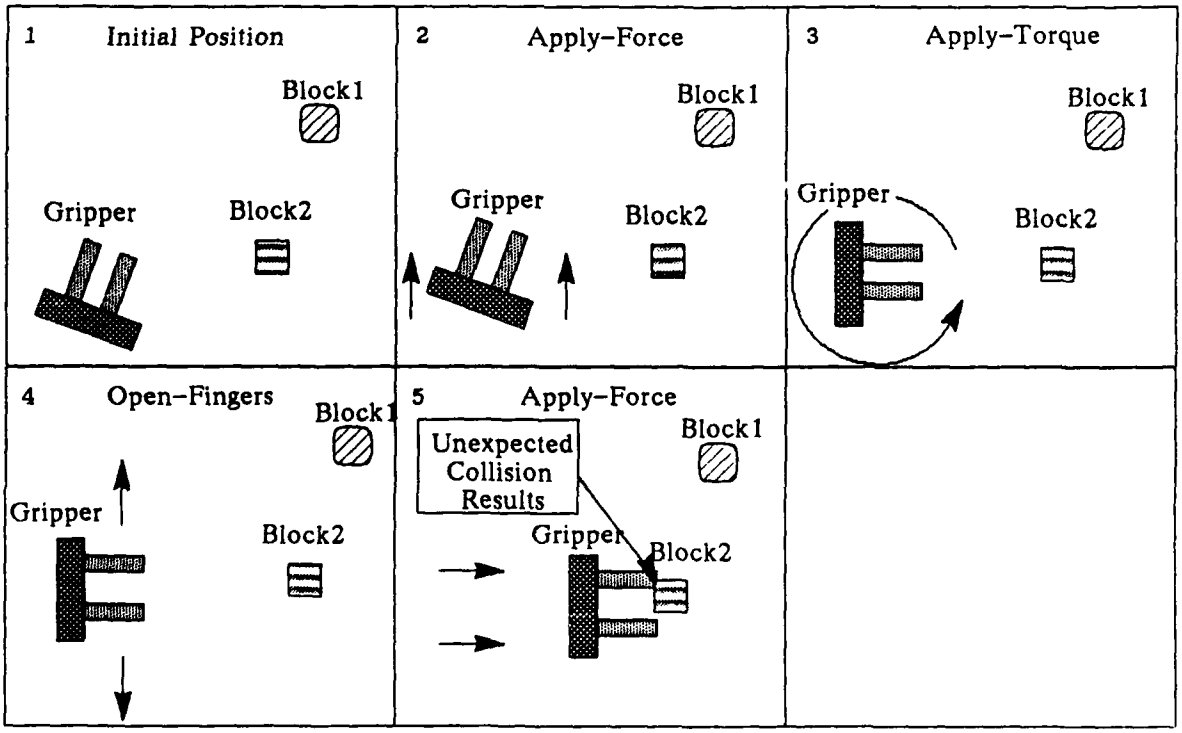


(4.8b) Real-World Precise View of the Operator Sequence

Figure 4.8. System's Initial Observations of a Grasp Being Performed



(4.9a) System's Approximate View of the Operator Sequence



(4.9b) Real-World Precise View of the Operator Sequence

Figure 4.9. System's First Attempt at Achieving a Grasp

initially observed plan did up until the *open–fingers* operation. Here, since the system uses only an approximate view of the data, it can't show the necessity for opening the gripper fingers wider than the block. However, the approximation lends itself to efficient reasoning about and execution of the plan. Unfortunately, in this case, these actions lead to a failure as shown in the precise view of Figure 4.9. Since the block was actually positioned slightly differently, the upper–left edge protruded, making contact with the gripper finger in an unexpected place. The system, only expecting a small contact force on the insides of the fingers, has a violation of expectations.

GRASPER now attempts to explain the failure by explaining the discrepancy between the real sensor readings and the expected ones. The system arrives at four primary single–fault failure explanations in this specific example which are rooted in possible errors in approximations for:

(1) block1's shape

(2) block1's position

(3) block2's shape

(4) block2's position

In this case, the only approximations not identified as possible causes of the failure are the orientation approximations. This is because no orientation error of either block could explain the encountered contact.

In general, failed approximations receive a likelihood rating in accordance with the degree to which the approximation needs to be changed to account for the failure. In this case, failures receive likelihood ratings inversely proportional to the distance the contact was from their expected positions. This is due to the method for position and shape approximations used for objects, which is illustrated in Figures 4.7 and 4.10, respectively. A threshold is used to avoid consideration of more unlikely causes for the failure. Here, block1's expected position is substantially farther from the point of contact than block2's position. Explanations corresponding to 1 and 2 in the list above are below threshold and are not considered.

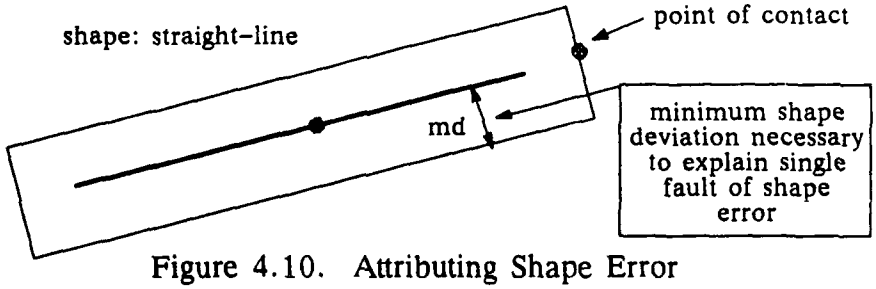


Figure 4.10. Attributing Shape Error

GRASPER's focusser suggests methods which the understander can use to focus on possible shape and position discrepancies with block2. Specifically, this is to increase the uncertainty margin considered in the approximate attributes of block2 to allow the understander to recognize fixes for the failures. The amount which the uncertainty margin is increased is a function of the attribution procedure illustrated in Figures 4.7 and 4.10 for position and shape. The approximation for an object is revised as little as necessary through consideration of each individual line segment that makes up its polygonal approximation. Having encountered a failure, the system now has a good

reason to invest more resources in trying to understand possible discrepancies in this area.

The understander constructs explanations for a new observed grasp operation from each of the failure perspectives. The generalizer generalizes each of these explanations and analyzes the resulting rules. In attempting to understand a successful grasp operation from each of the failure perspectives, each of the failures actually was determined to be a motive in opening the gripper wider. That is, opening wider and closing actually can take care of small variations in shape and positional uncertainty of the object being grasped. In the post–generalization analysis, after the system has performed some rule simplification, it is found that one of the rules actually deals with both of the potential failures and is thus more general. Although this may not always be the case, the system is capable of recognizing such occurrences and profiting from them. Naturally, that rule is used in the knowledge base and the system now has a good method for dealing with two types of uncertainty in this type of grasp operation.

5 UNDERSTANDING OBSERVED MANIPULATOR SEQUENCES

Central to any explanation–based learning system is the method by which more efficient plans are discovered. This process is known as *understanding* and is similar to methods used in natural language understanding.

5.1 Basic Elements of the System's Knowledge Representation

It is necessary that a few aspects of the system's knowledge representation be discussed now so as to clarify the techniques to be presented later in the section. The system assumes a complete albeit intractable set of domain knowledge which equips it to deal with real–world robot manipulation situations. The domain knowledge is organized into *facts*, *rules*, and *procedures*. Facts and rules are encoded in a first–order predicate calculus representation which can be examined by the system. We therefore refer to these components of the domain knowledge as *introspectable*. The procedural knowledge represents low–level built–in functions such as the basic arithmetic operators which need to be fast and have internal logic which it is unnecessary to have the system to reason about. The procedural knowledge consists of actual program code and is *nonintrospectable*.

Facts are represented as ground instances. For instance, a fact might relate the initial Cartesian X coordinate of a certain robot gripper as

```
(gripper-x gripper1 -7.5).
```

Actions the system takes serve as transitions between *situations:* complete specifications of world state. Facts are therefore true in a specified situation or situations.

Rules are a general structure for representing inference rules and actions. The general form for a rule is

```
(rule :cons <consequent 1> <consequent 2> ... <consequent n>
      :ants <antecedent 1> <antecedent 2> ... <antecedent n>).
```

Rule semantics are such that the conjunct of the antecedents implies the conjunct of the consequents. Disjunctions are not allowed in rules. The system uses an approach similar to the STRIPS formalism of add and delete lists [Fikes72]. Consequents are predicate expressions just as were the facts discussed above. Consequents of the form *(not <predicate>)* are handled as if <predicate> were on a delete list. That is, *<predicate>* would be true in the situation prior to the rule application but not true in the following situation. Antecedents are also of predicate form and can be proved true through unification with facts, procedures, and/or through unification with and backward chaining on the consequents to other rules.†

Several of the predicates illustrated in the example in this chapter should be defined. They are

choose–grip

given a gripper and an object this predicate determines a target position and angle from which to grasp the object

---

† Implementation Note: It is interesting to note that some have approached the representation of domain knowledge in Prolog [Prieditis87]. Prolog does provide an obvious representation for rules of Horn clause form. Beyond that, representation of more complex rules necessitates more awkward higher level work on top of the basic Prolog system. It also seems to make search control more difficult. It is for these reasons that the system was written in Common Lisp.

forall

checks that a set of conditions are true for each member of a list with the exception of those for which an exception condition evaluates to true

gripper-approach-sequence

finds a gripper approach sequence which includes choosing an approach position, angle, and opening width and gripping position and angle

gripper-clear-translate

checks whether it is possible to translate the gripper from one position to another free of collisions

translate-poly-area

returns as a polygon the area traversed by the gripper during a translate

world-objects

returns a list of the objects in the world

Consider the following rule used in the system:

```
(rule :cons (gripper-clear-translate ?gripper ?current-x ?current-y ?target-x
                                     ?target-y ?object-list ?gripper-beam-length
                                      ?gripper-beam-width ?gripper-finger-length
                                      ?gripper-finger-width
                                      ?gripper-finger-separation ?gripper-angle)
      (gripper-x ?gripper ?target-x)
      (gripper-y ?gripper ?target-y)
      (not (gripper-x ?gripper ?current-x))
      (not (gripper-y ?gripper ?current-y))
```

```
:ants (translate-poly-area ?gripper ?current-x ?current-y ?target-x
                           ?target-y ?polygon ?gripper-beam-length
                           ?gripper-beam-width ?gripper-finger-length
                          ?gripper-finger-width ?gripper-finger-separation
                           ?gripper-angle)
      (world-objects ?object-list)
      (forall ?name ?object-list
              ((not (equal ?name ?gripper)))
              (not (intersect-object-poly ?name ?polygon))))
```

This rule represents a gripper translation that is completely clear of contact with other objects in the world. In addition to achievement of the *gripper-clear-translate* goal, other consequents include the change in position of the gripper. Therefore, the current gripper position is marked for deletion and the new position is asserted. The antecedents of this rule consist of predicates which calculate a polygonal representation of the area covered by the translation, enumerate objects the system knows to exist, and iterate over a list establishing that none of the known objects interfere with the translation path.

An explanation for a goal achievement has three primary components: a list of rules used in achieving the goal, a list of consequent antecedent pairs where the antecedent of one rule used in achieving the goal was satisfied by the consequent of another,

and a variable binding list. The binding list gives values for rule variables as they appeared in the specific goal achievement.

5.2 The Suggestion Mechanism

The sequence an understander observes consists only of robot manipulator primitives. The understander must somehow attribute purpose to these observed primitives. The most natural way to do this is by a technique known as *suggestion*. For example, if a *close–fingers* primitive is observed one suggestion might be that a *grip* was being attempted. This is by no means a certainty. Naturally, the gripper fingers could have been closed for many other reasons. Maybe the fingers were closed because they were previously open too far for an optimal approach to an object. That is, they unnecessarily increased the risk of striking an object on the approach. Therefore, suggestion is used here as a heuristic function for what goal likely produced the observed effect.

If the system knew all the ways in which the knowledge would be applied to problem solving, an analysis of the knowledge could produce an optimal set of suggestion heuristics. Unfortunately, such a complete analysis is intractable. We use a practical approach as illustrated with the suggestion rule of Figure 5.1 which is built into the understander. This rule states that good suggestions are ones in which the antecedent suggesting a particular consequent is found as an antecedent relatively infrequently in the rule set. The threshold τ used in the rule indicates a frequency above which an antecedent becomes a poor predictor. The threshold can be dynamically tuned, based on the performance of the understander. When the threshold is too low, the understander fails

to understand due to a lack of suggestions. When the threshold is too high, the understanding process takes much more processing time and can become unmanageable.

Given:

a set of rules σ
nodes α and β
a numeric threshold τ

If:

$\rho \in \sigma$
α is an antecedent of ρ
β is a consequent of ρ
α appears as an antecedent in less than τ rules over σ

Then:

α suggests β

Figure 5.1. Suggestion Rule

5.3 An Algorithm For Understanding

An algorithm for understanding is illustrated in Figure 5.2. Initially, a queue is built with entries representing the observations the system is seeking to understand. The goal is to produce a series of connected explanations for these observations. Each node on the queue is examined in turn. First, a list of nodes which it suggests are identified. For each of these we attempt to prove that the suggested node could have taken place. If any of the suggested nodes could have taken place, they are added to the end of the queue. Otherwise, the node itself is replaced on the end of the queue as its suggestions may later become provable through the integration of information provided by the other

```
queue <- a nonempty list of observations in node form

1:
if an element of queue satisfies a primary system goal then
begin
  result <- queue
end
else
begin
    foreach node in queue do
    begin
        tail-queue <- nil
        suggested-nodes <- list of nodes suggested by node
        a-suggestion-worked <- nil
        foreach suggested-node in suggested-nodes do
        begin
            this-suggestion-worked <- nil
            foreach rule of which suggested-node is a consequent do
            begin
                urule <- a unique copy of rule
                foreach antecedent a in urule do
                begin
                    unless this-suggestion-worked do
                        try to prove the conjunct of urule's antecedents under the binding set
                        produced by the unification of node with the corresponding urule
                        antecedent
                        if the antecedents could be proved then
                        begin
                            this-suggestion-worked <- t
                            a-suggestion-worked <- t
                            assert that the consequents of the rule are thought to have occurred
                            (update add/delete lists accordingly)
                            add the previously suggested consequent of urule to the tail-queue
                        end
                end
            end
        end
        unless a-suggestion-worked add node to the tail-queue
    end
    if a-suggestion-worked then
    begin
        queue <- tail-queue
        goto 1
    end
    else result <- queue
end
```

Figure 5.2. An Understanding Algorithm

observations. This process continues until no further suggestions can be confirmed from the nodes on the queue or a node on the queue is discovered to satisfy one of the system's key goals. After having eliminated equivalent explanations, all the elements on the queue then represent partial explanations as to what might have taken place. Usually, in observing intelligent behavior, the set of observations will have been reduced to explanations which are able to explain all of the observations rather than just a few.

The differences between this understander and those in standard explanation–based learning system lie primarily in the domain knowledge used in the understanding. That is, some of the domain knowledge is approximate. This has no effect on the functionality of the understander. Another difference, however, is that the understander interprets things according to the approximate world model. This leads to the possibility of interpreting some aspects of observed behavior as unnecessary.

The system's operationality criteria should be used to determine the interpretation given to the observations. For instance, in our example, the human operator of the robot gripper is observed to open much wider than the piece being grasped. Any physically possible opening width greater than the width of the piece would satisfy the goal under the system's world model. The system then faces a complicated operationality tradeoff in selecting the appropriate width to use. Opening very wide may improve uncertainty tolerance but may sacrifice generality as in many situations other objects nearby may prohibit such a wide opening. Movement of the gripper's fingers is part of the economy of execution of the plan. A desire purely for economy of execution would

favor moving the gripper's fingers the least possible from their current position such that they are open wider than the piece. In our implementation, the preferred opening width is to open just wide enough to clear the object as described in the system's world model.

5.4 An Example

The understander starts with a sequence of observed actions as follows:

```
OBSERVED-APPLY-FORCE
  GRIPPER1 <---------- Gripper Name
  5 <----------------- Motor Force Applied
  0 <----------------- Numeric Starting Time
  1 <----------------- Numeric Ending Time
  -7.5 <-------------- Current Gripper X-Coordinate
  8.5 <--------------- Current Gripper Y-Coordinate
  -9.5 <-------------- Target X-Coordinate
  3.5 <--------------- Target Y-Coordinate
  4 <----------------- Gripper Beam Length
  2 <----------------- Gripper Beam Width
  3 <----------------- Gripper Finger Length
  2 <----------------- Gripper Finger Width
  2 <----------------- Current Gripper Finger Separation
  -90 <--------------- Current Gripper Angle

OBSERVED-APPLY-TORQUE
  GRIPPER1 <---------- Gripper Name
  5 <----------------- Motor Force Applied
  1 <----------------- Numeric Starting Time
  2 <----------------- Numeric Ending Time
  -90 <--------------- Current Gripper Angle
  0 <----------------- Target Gripper Angle
  4 <----------------- Gripper Beam Length
  2 <----------------- Gripper Beam Width
  3 <----------------- Gripper Finger Length
  2 <----------------- Gripper Finger Width
  2 <----------------- Current Gripper Finger Separation
  -9.5 <-------------- Current Gripper X-Coordinate
  3.5 <--------------- Current Gripper Y-Coordinate

OBSERVED-OPEN-FINGERS
  GRIPPER1 <---------- Gripper Name
  5 <----------------- Motor Force Applied
  2 <----------------- Numeric Starting Time
  3 <----------------- Numeric Ending Time
```

```
  2 <---------------- Current Gripper Finger Separation
  3.75 <------------- Target Gripper Width
  -9.5 <------------- Current Gripper X-Coordinate
  3.5 <-------------- Current Gripper Y-Coordinate
  0 <---------------- Current Gripper Angle
  2 <---------------- Current Gripper Finger Separation
  2 <---------------- Gripper Beam Width
  4 <---------------- Gripper Beam Length
  2 <---------------- Gripper Finger Width
  3 <---------------- Gripper Finger Length

OBSERVED-APPLY-FORCE
  GRIPPER1 <---------- Gripper Name
  5 <---------------- Motor Force Applied
  3 <---------------- Numeric Starting Time
  4 <---------------- Numeric Ending Time
  -9.5 <------------- Current Gripper X-Coordinate
  3.5 <-------------- Current Gripper Y-Coordinate
  4.5 <-------------- Target X-Coordinate
  3.5 <-------------- Target Y-Coordinate
  4 <---------------- Gripper Beam Length
  2 <---------------- Gripper Beam Width
  3 <---------------- Gripper Finger Length
  2 <---------------- Gripper Finger Width
  3.75 <------------- Current Gripper Finger Separation
  0 <---------------- Current Gripper Angle

OBSERVED-CLOSE-FINGERS
  GRIPPER1 <---------- Gripper Name
  5 <---------------- Motor Force Applied
  4 <---------------- Numeric Starting Time
  5 <---------------- Numeric Ending Time
  3.75 <------------- Current Gripper Finger Separation
  3 <---------------- Target Gripper Width
  4.5 <-------------- Current Gripper X-Coordinate
  3.5 <-------------- Current Gripper Y-Coordinate
  0 <---------------- Current Gripper Angle
  3.75 <------------- Current Gripper Finger Separation
  2 <---------------- Gripper Beam Width
  4 <---------------- Gripper Beam Length
  2 <---------------- Gripper Finger Width
  3 <---------------- Gripper Finger Length
```

A node is created for each of these actions and is placed on a queue. In the next phase, the system attempts to suggest various higher-level goals of which each of the observed actions may be a part. It will then try to follow these up in a top-down

fashion. The first *observed–apply–force* suggests two possible rules of which it may be a part. Not surprisingly, both are versions of *apply–force*. However, each of the versions of *apply–force* carry along with them different expectations on the part of the operator. By associating the bottom–level *observed–apply–force* with one of these, the system is beginning to infer goals on the part of the external agent. The two versions of *apply–force* are shown below. The first *apply–force* is used as the final approach to an object which we desire to grip. It carries the expectation of contacting the goal object in a gripping position. The second is used as an unimpeded translate from one location to another. It carries the expectation that no objects will interfere with the object being moved during the translate.

```
(rule :cons :nc (apply-force ?gripper ?dx ?dy ?force ?start ?end
                  (gripper-approach-sequence ?gripper ?dx ?dy ?object ?face1
                                             ?face2 ?mid-x ?mid-y ?angle
                                             ?separation ?gx ?gy ?ob-poly ?gcx
                                             ?gcy ?ob-list ?gbl ?gbw ?gfl ?gfw
                                             ?angle)
                  (at-coordinates ?mid-x ?mid-y ?gripper)
                  ((gripper-x ?gripper ?mid-x)
                   (gripper-y ?gripper ?mid-y)
                   (gripper-angle ?gripper ?angle)
                   (gripper-finger-separation ?gripper ?separation)))
      :ants (observed-apply-force ?gripper ?force ?start ?end ?gx ?gy ?mid-x ?mid-y ?gbl
                                  ?gbw ?gfl ?gfw ?separation ?angle)
            (world-objects ?ob-list)
            (dif ?dx ?mid-x ?gx)
            (dif ?dy ?mid-y ?gy)
            (choose-grip ?gripper ?object ?mid-x ?mid-y ?angle ?min-separation ?face1
                                  ?face2 ?perror ?oerror)
            (less-than-or-equal ?min-separation ?separation))
```

```
(rule :cons :nc (apply-force ?gripper ?dx ?dy ?force ?start ?end
                    (gripper-clear-translate ?gripper ?cx ?cy ?tx ?ty ?ob-list
                                             ?gbl ?gbw ?gfl ?gfw ?gfs ?ga)
                    (at-coordinates ?tx ?ty ?gripper)
                    ((gripper-x ?gripper ?tx)
                     (gripper-y ?gripper ?ty)
                     (gripper-angle ?gripper ?ga)
                     (gripper-finger-separation ?gripper ?gfs)))
      :ants (observed-apply-force ?gripper ?force ?start ?end ?cx ?cy ?tx ?ty ?gbl ?gbw
                                  ?gfl ?gfw ?gfs ?ga)
            (world-objects ?ob-list)
            (dif ?dx ?tx ?cx)
            (dif ?dy ?ty ?cy))
```

Initially, the first action of *observed–apply–force* unifies successfully with the like term in the first of our two rules. Using this binding list, an attempt is made to prove the antecedents of that rule in order to infer it had taken place in the external agent's planning process. At this stage, the specifically bound antecedents to the rule could be written as

```
(observed-apply-force gripper1 0 0 1 -7.5 8.5 -9.5 3.5 4 2 3 2 2 -90)
(world-objects ?ob-list)
(dif ?dx -9.5 -7.5)
(dif ?dy 3.5 8.5)
(choose-grip gripper1 ?object -9.5 3.5 -90 ?min-separation ?face1
?face2                                    ?perror ?oerror)
(less-than-or-equal ?min-separation 2).
```

The first is trivial as it was an observed action and was already unified with the antecedent. The *world–objects* predicate looks up the known world objects as this information is part of the rule consequents. The next two *dif* (subtraction) predicates calculate relative move coordinates for use in the consequents. The *choose–grip* predicate indicates that this rule is designed for use of a*pply–force* while anticipating contact with an object to be gripped. Without going into the subproof for *choose–grip* here (see the appendix for the complete example), the *choose–grip* could only succeed if there were an object to grasp at the end point (–9.5 3.5) of the *apply–force*. There is no such object at that location. Consequently, the *choose–grip* fails and this suggested use of *apply–force* will not be confirmed.

The second rule, on the other hand, does apply.. It makes no special conditions, at this point, about there being an object at the termination of the *apply–force.* The antecedents for the second rule after the unification of the *observed–apply–force* could be instantiated as follows:

```
(observed-apply-force gripper1 0 0 1 -7.5 8.5 -9.5 3.5 4 2 3 2 2 -90)
(world-objects ?ob-list)
(dif ?dx -9.5 -7.5)
(dif ?dy 3.5 8.5)
```

All of these antecedents can be proved so the consequents of the rule are asserted. In the next level of suggestion, in the example, expectations of the inferred *apply–force*

will be confirmed. This is an expectation of an object–free path from initial to final positions of the *apply–force*.

The following shows how the queue appears at each cycle of the algorithm until one of the nodes, having arrived at *grasp*, has achieved a basic system goal.

```
(0) Queue: observed-apply-force observed-apply-torque
           observed-open-fingers observed-apply-force
           observed-close-fingers

(1) Queue: apply-force apply-torque open-fingers apply-force close-fingers

(2) Queue: move-gripper-clear rotate-gripper-clear adjust-fingers
           approach-target-clear grip

(3) Queue: move-for-approach rotate-for-approach
           adjust-fingers-for-approach approach-target
           achieve-frictional-force

(4) Queue: prepare-for-approach prepare-for-approach prepare-for-approach
           achieve-frictional-force grasp
```

Stage 0 shows the initial contents of the queue containing only the observed actions, no inferences having yet been made. Next, primitive actions and expectations are inferred. Eventually, in steps 3 and 4, the first three observed actions are recognized to be part of preparing for an approach to an object. All along, the *close–fingers* action

provided the strongest clue to the goal of the operation. From it was inferred that an object was being gripped and that a frictional force was being sought between the gripper and an object. This indicated a *grasp* which could suggest no further rules and which actually made use of all the observed actions. An abbreviated view of the final explanation structure is shown in Figure 5.3. The reader is referred to the appendix where a complete demonstration of the understander on this example is documented.

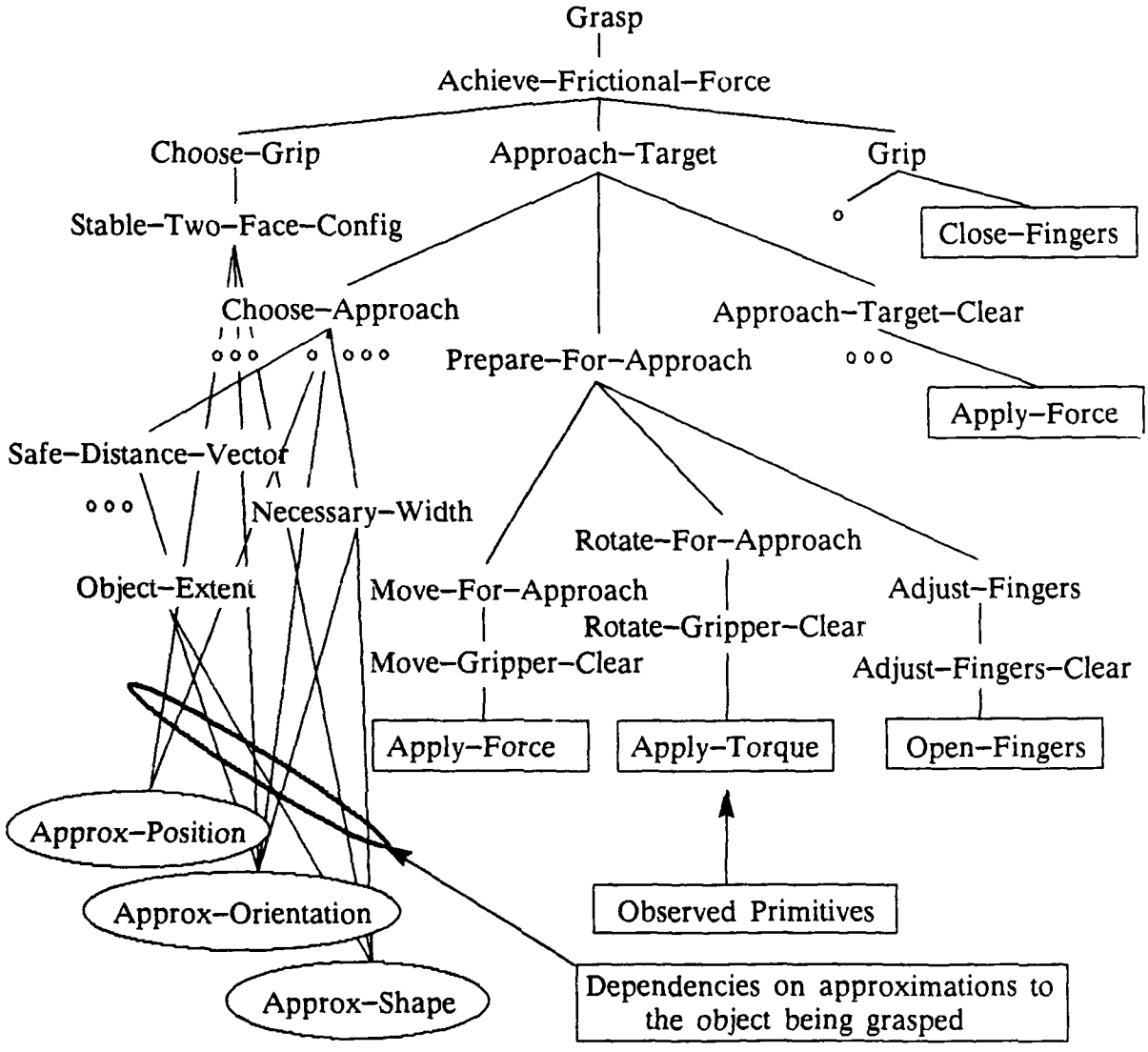


Figure 5.3. An Abbreviated View of the Explanation Structure

6 EXPLAINING AND USING FAILURES

The first phase of dealing with failures involves identifying when they have occurred. The executive component is responsible for this. Whenever a primitive is executed, a prediction is generated based on the domain knowledge, world model, and current state as to the values of several measurable parameters. These parameters are compared to those values measured from the real–world environment to provide feedback on the success of the action. In the current implementation, since no interface has yet been constructed with a real robot arm, the measurements are obtained through a world simulator.

Before discussing the specifics of the mechanism for learning from failures in the robotics domain, it is necessary to describe the approximations used to represent uncertainty of world objects. Physical objects such as the gripper, about which we assume perfect knowledge as discussed earlier, are represented by their exact polygons. All other physical objects are approximated through use of position, orientation, and shape approximations. Each of the three basic approximations includes the attribute and update procedures. For a position approximation, the attribution function measures the minimum distance from a point of contact to the polygon representing the object (see Figure 6.1). The associated update function extends the uncertainty margin to include the possibility that the contact point was on the object. The functions for the object orientation approximation are similar. Instead of using distance to the object, the minimum angle by which the polygon representing the object could be rotated to produce the

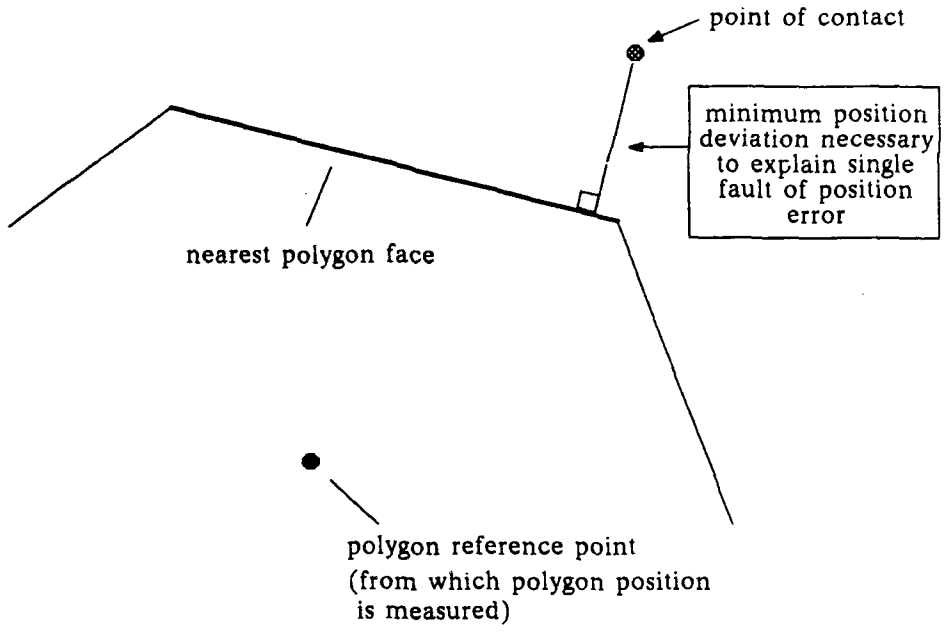


Figure 6.1. Attributing Position Error

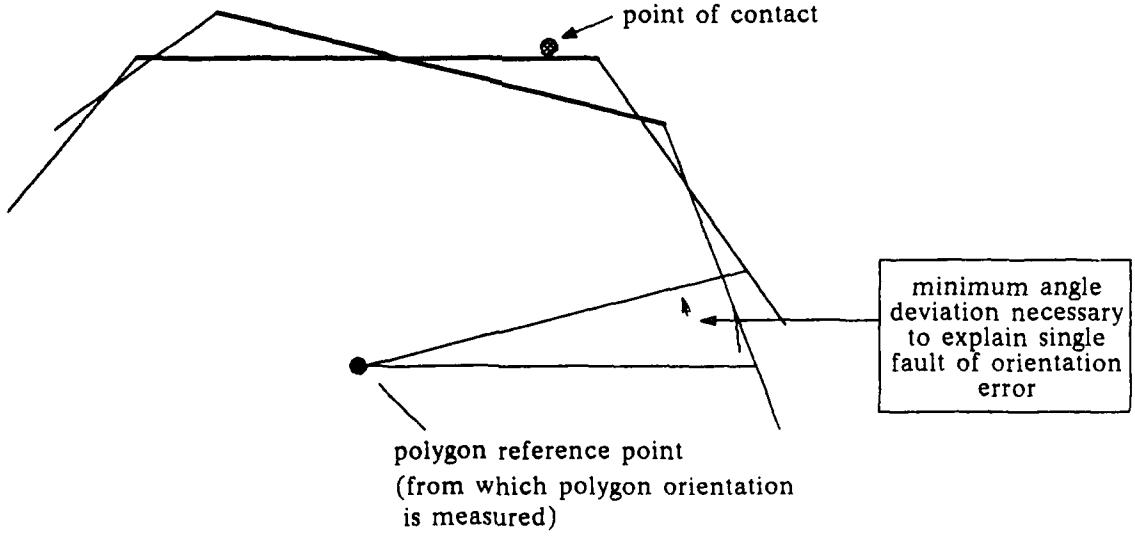


Figure 6.2. Attributing Orientation Error

encountered contact is measured for the attribution function (Figure 6.2). The update function extends the angular uncertainty to account for such a contact with the object. Last, the object shape approximation measures the minimum deviation for a face of the

polygon such that the contact could be achieved (Figure 6.3). The update function

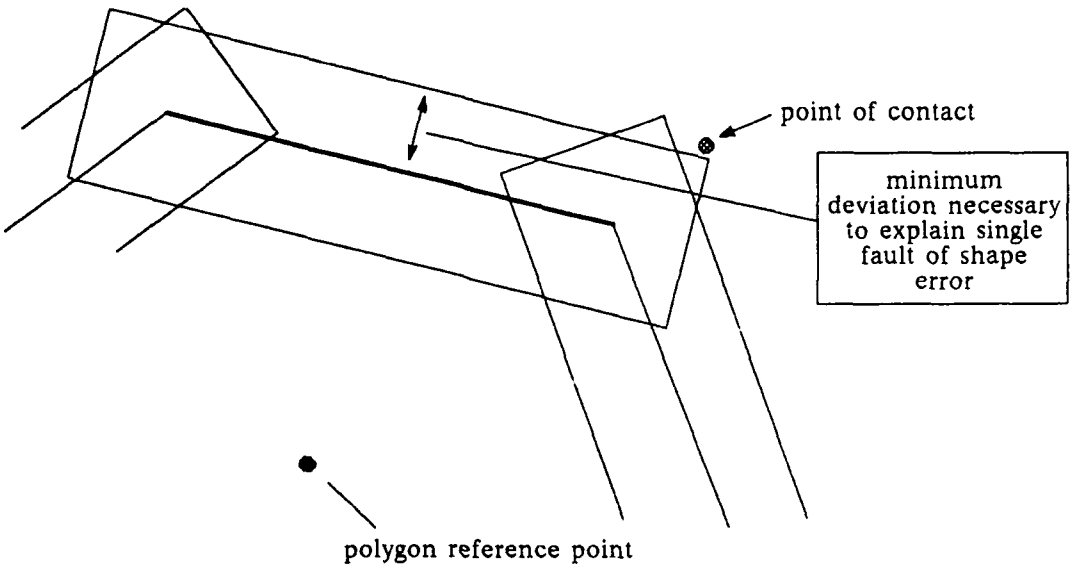


Figure 6.3. Attributing Shape Error

extends the shape error of each side of the polygon to account for the contact.

The system's executive component performs actions moving through a series of qualitatively different states. During each action, the system has a set of sensory expectations. The actual sensor values are monitored to see that they lie within expected limits. When expectations are violated, a failure is said to have occurred. In the example introduced in Chapter 4, such a violation occurs as the gripper strikes an object slightly before it was expecting to contact the object it was attempting to surround for the grasp. Figure 6.4 illustrates the situation at the time of the failure. The executive indicates the failure and returns information about the expected sensor values as follows:

```
Prior Value Dump:
                    (GRIPPER-X GRIPPER1 -8.5)
```

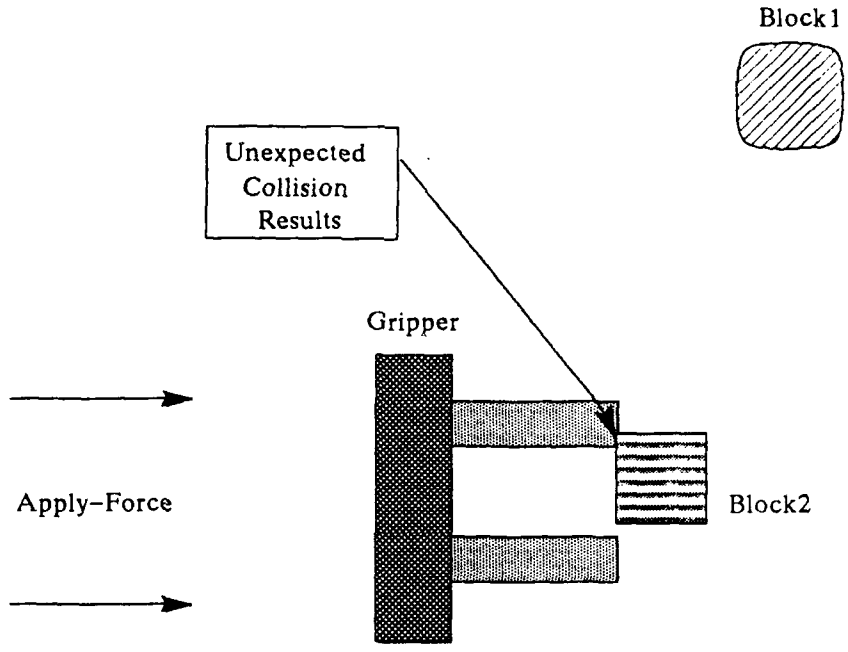


Figure 6.4. Illustration of a Failure

```
                    (GRIPPER-Y GRIPPER1 3.5)
                    (GRIPPER-ANGLE GRIPPER1 0)
                    (GRIPPER-FINGER-SEPARATION GRIPPER1 3.0)
                    (NOT (GRIPPER-X GRIPPER1 -7.5))
                    (NOT (GRIPPER-Y GRIPPER1 8.5))
                    (NOT (GRIPPER-ANGLE GRIPPER1 -90))
                    (NOT (GRIPPER-FINGER-SEPARATION GRIPPER1 2))
Dump At Failure:
                    (GRIPPER-X GRIPPER1 1.5)
                    (GRIPPER-Y GRIPPER1 3.5)
                    (GRIPPER-ANGLE GRIPPER1 0)
                    (GRIPPER-FINGER-SEPARATION GRIPPER1 3)
                    (FORCE GRIPPER1 FINGER1 3 0.25 5 -1 0)
                    (NOT (GRIPPER-X GRIPPER1 -7.5))
                    (NOT (GRIPPER-Y GRIPPER1 8.5))
                    (NOT (GRIPPER-ANGLE GRIPPER1 -90))
                    (NOT (GRIPPER-FINGER-SEPARATION GRIPPER1 2))
Expectation Goal:
  GRIPPER-APPROACH-SEQUENCE
    GRIPPER1 <---------- Gripper Name
    13.0 <-------------- Delta X
    0.0 <--------------- Delta Y
    SQUARE2 <----------- Object
    F1 <---------------- One Face Of The Object
    F3 <---------------- Another Face Of The Object
    4.5 <--------------- Target X-Coordinate
```

```
    3.5 <--------------- Target Y-Coordinate
    0 <----------------- Target Gripper Angle
    3.0 <--------------- Target Gripper Width
    -8.5 <-------------- Current Gripper X-Coordinate
    3.5 <--------------- Current Gripper Y-Coordinate
    #<Structure POLY 10A640CB> <-- Polygon Representation Of The Object
    1.5 <--------------- Nearest Gripper X-Position Prior To Object Contact
    3.5 <--------------- Nearest Gripper Y-Position Prior To Object Contact
    (SQUARE1 GRIPPER1 SQUARE2)
    4 <----------------- Gripper Beam Length
    2 <----------------- Gripper Beam Width
    3 <----------------- Gripper Finger Length
    2 <----------------- Gripper Finger Width
    0 <----------------- Current Gripper Angle
Expectation Expl:
  #<Structure EXPLANATION 10AF49F3>
Expected Dump:
                    (GRIPPER-X GRIPPER1 1.5)
                    (GRIPPER-Y GRIPPER1 3.5)
                    (GRIPPER-ANGLE GRIPPER1 0)
                    (GRIPPER-FINGER-SEPARATION GRIPPER1 3)
                    (NOT (GRIPPER-X GRIPPER1 -7.5))
                    (NOT (GRIPPER-Y GRIPPER1 8.5))
                    (NOT (GRIPPER-ANGLE GRIPPER1 -90))
                    (NOT (GRIPPER-FINGER-SEPARATION GRIPPER1 2))
```

The discrepancy between the expected and actual dumps at the time of failure lies with the presence of an unexpected force. The failure explainer evaluates each of the object approximations utilized in the plan with respect to the unexpected contact position (see Figure 6.5). This is done using the attribute function for each approximation. While the algorithm currently employed does look at all approximations, future plans include only considering those approximations likely to have contributed to the failure (e.g., using a distance threshold). Of the six approximations evaluated (position, orientation, and shape of each of two squares), only failures in the position and shape approximations could have resulted in the observed failure. These four possible contributory approximations are passed through a plausibility filter which eliminates ap-

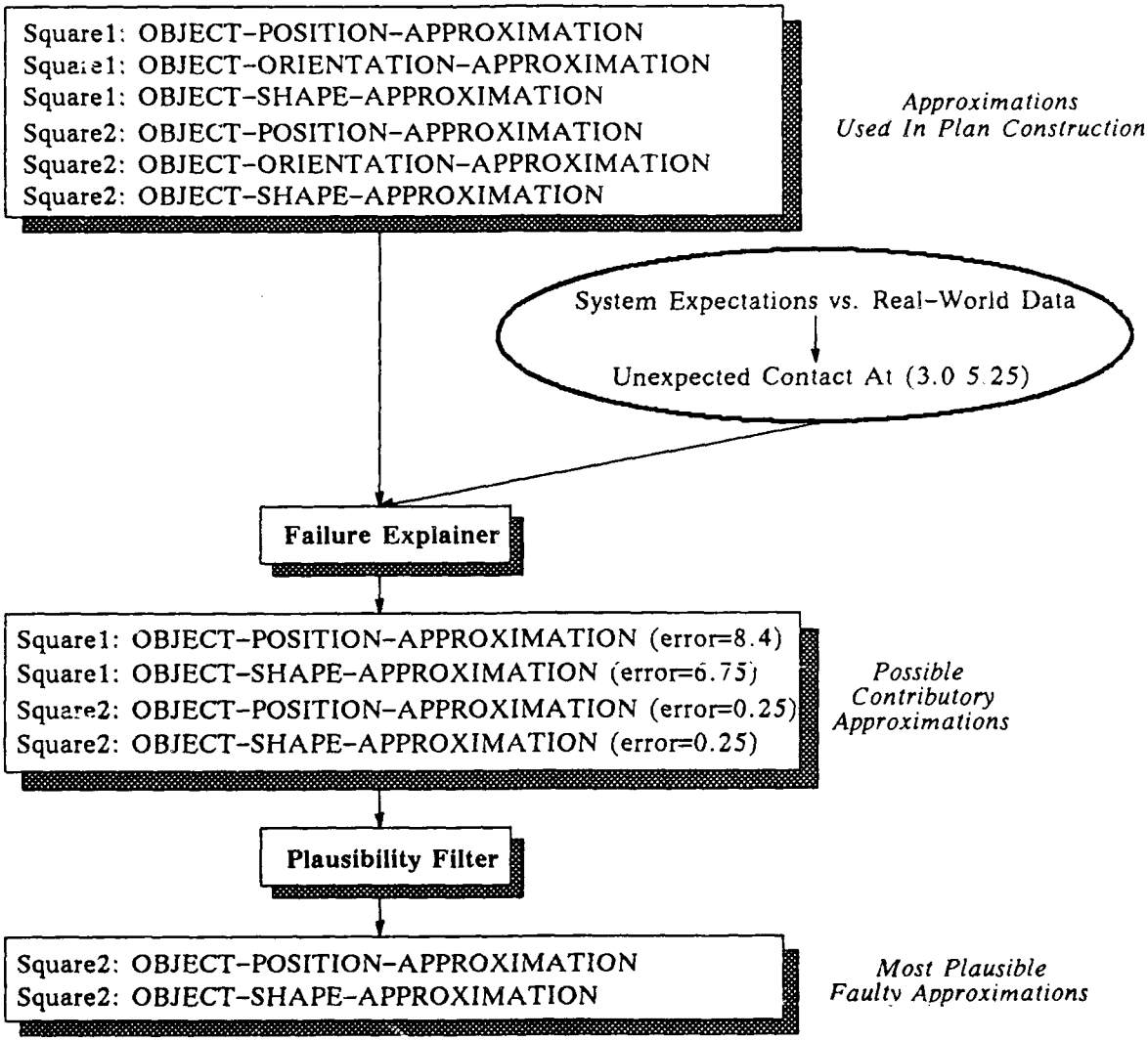


Figure 6.5. Narrowing a Set of Candidate Faulty Approximations

proximations from consideration whose error would have to be very large to explain the failure. In this case, the approximations to Square1 are eliminated as their errors would have to be very large to explain the contact near Square2 and are, therefore, over a predetermined plausibility threshold. Figure 6.6 shows a comparison of errors for the position approximations for squares 1 and 2. If the position approximation for square 1 were to blame for the failure, it must have been grossly in error, which is inconsistent

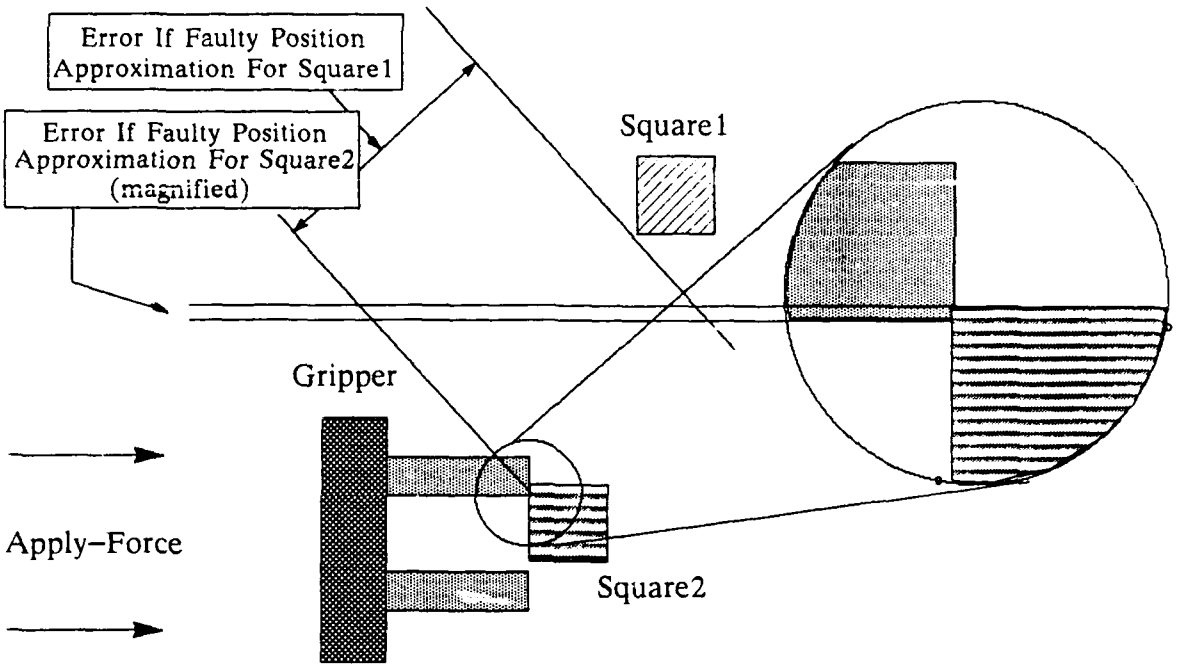


Figure 6.6. A Comparison of Errors for Faulty Position Approximations of Squares 1 and 2

with the system's basic assumption that uncertainties tend to be small. The remaining two most plausible faulty assumptions are input to the tuning selector for analysis.

The tuning selector retrieves the initial primitive operator sequence that led to construction of the failed plan and produces a set of plans each of which results from tuning one of the approximations being considered. An approximation is tuned using its update function which, in this case, makes use of the position of the unexpected contact. Under the assumption that only one faulty approximation caused the failure, at least one of the produced plans will remedy the problem. The tuning selector gives preference to the most operational plan. Measuring operationality of plans is an area of ongoing work (see Chapter 8). The current GRASPER implementation uses a technique as follows: First, each of the plans is simplified to aid in comparing them. Simplification occurs

when some of a plan's antecedents can be satisfied independent of context. For instance, consider an antecedent which consists of the following predicate:

(sum ?x11 3 4)

The binding for ?x11 (7) can be calculated prior to application of the plan. The constant 7 is then substituted for all occurrences of ?x11 throughout the plan. The simplification process involves repeatedly performing such calculations until no further reductions can be made. Once simplification is complete, the plans are compared to each other. The current comparison technique compares plans with the same consequents and action sequences. If plan A's antecedents are a subset of plan B's, plan A is preferred to plan B in coverage as it will also remedy the approximation failure which led to creation of plan B. A full operationality comparison is planned in the future.

In the example, the plans produced from tuning of Square2's position and shape approximations are already in simplified form and are totally comparable. Both specify that the gripper should be opened 0.25 unit wider than in the original plan. One of the two plans is therefore picked arbitrarily to replace the original plan and the tuned approximation which generated it is now utilized in the world model.

This approach demonstrates how explicit object approximations can be reasoned about at failure recovery time so as to produce a set of candidate revised plans the most operational of which is selected. If the approximations don't deviate widely from their targets in reality and the operationality criterion is suited to the system and environment, this method will take full advantage of the information gained from each failure. Fur-

thermore, the revised plan is designed to tolerate uncertainty rather than to restrict its application so as to avoid it.

7 RELATED WORK

In this chapter, related work is discussed in the areas of: machine learning and robotics, reasoning about uncertainty in robotics, learning from failures, approximation and learning, and operationality.

7.1 Learning and Robotics

One of the earliest systems utilizing machine–learning techniques for controlling a robot was the STRIPS system [Fikes72]. It was able to learn macro–operators for operating a robot from analysis of its own problem solving. The generalization technique used for constructing the macro–operators is very similar to the EGGS technique used in explanation–based learning today.

The first work in demonstrating the application of explanation–based learning (EBL) to problem solving in robotics was done by Segre in his ARMS system [Segre87]. ARMS observes a human operator achieving some goal through sending commands to a robot manipulator. Using its domain theory, the system is able to construct a general plan for achieving the goal. The plan is not sensitive to incidentals in the original training example as are plans generated using other learning techniques. This is because only aspects of the training example which support the goal, according to the domain theory, find their way into the final plan.

The problem with ARMS is that it uses an idealized model of the world as other current EBL systems do. The world is never ideal like the model and ARMS has no way of reasoning about possible disparities between the model and real world. Furthermore,

its plans will never incorporate uncertainty tolerance because of these representational shortcomings.

Mel's MURPHY system is a first step in using a connectionist learning technique to aid robot manipulation [Mel88]. MURPHY uses knowledge about a robot arm's current joint configurations in conjunction with visual data about the joint configurations to learn connections between the two. This can be accomplished without an intelligent teacher by having the robot step through a representative sample of the 1 billion possible joint configurations. MURPHY can use its learned connections to "envision" sequences of actions for planning.

MURPHY is quite appealing as a method for learning sensory–motor interaction in a robot arm because of its simplicity. It requires little domain knowledge but faces the common disadvantages of such connectionist techniques: it requires a myriad of examples, needs an environment to "train" in where failures have negligible cost, and with little domain knowledge can be sensitive to incidental associations made from observations.

Several systems using similarity–difference–based learning (SDBL) have also been used for controlling robots [Andreae84, Whitehall87]. Whitehall's PLAND system observes a trace of robot activity and develops macro–operators which include loops, conditionals, and sequences. This system is designed to function with only a single positive example and little domain knowledge. Although the system can develop a

macro–operator based on the one example, with little domain knowledge, not much confidence can be expressed in the resulting operator. Systems like these help to demonstrate that learning systems are located on a knowledge continuum ranging from those systems very good at dealing with knowledge poor domains but requiring many examples like SDBL systems and those good at operating in knowledge rich domains with only a few examples like EBL systems. Eventually, these approaches will merge into one system to which the amount of available knowledge determines which learning techniques apply and the system can function comfortably anywhere on the continuum. GRASPER uses a rich domain theory and generates a plan from a single training example but also utilizes future training examples to improve the plan as failures are observed.

7.2 Reasoning About Uncertainty in Robotics

Brooks provides a logic for reasoning about uncertainties and defines a plan checker capable of recognizing when a plan cannot deal with a certain range of uncertainties and adding sensing constraints to ensure its success [Brooks82]. This approach propagates error ranges throughout the plan to infer what effects they will have. This is computationally an expensive technique. Our approach doesn't seek a complete error analysis so as to preserve system efficiency.

Brost provides a detailed algorithm for automatic grasp planning in the presence of uncertainty [Brost88]. For instance, this includes a detailed analysis of the squeeze–grasp, the grasp used by GRASPER in our examples. This grasp can succeed

in spite of some uncertainty in object position, shape, and orientation. Brost's approach, like Brooks', involves a detailed analysis of the grasp and the possible uncertainties to determine which grasp to use and exactly how it should be applied to succeed. Brost has made a number of assumptions in order to simplify the algorithm which handles three basic grasping methods. Brost's technique is expensive like Brooks and also has no provision for correcting itself. GRASPER uses observation to acquire its grasping techniques and uses approximations to promote tractability. Unlike the fixed algorithms of Brooks and Brost, GRASPER can tune approximations and regenerate the plan so as to adapt to observed failure situations.

7.3 Learning from Failures

The concept of using failures to trigger learning is not a new one. Many researchers have used incremental techniques for learning. One of the earlier works in this area was Sussman's HACKER system [Sussman73]. HACKER operated in a blocks–world, constructing plans and simulating their execution. The system initially assumed it could achieve subgoals independently. Naturally, this led to the generation of incorrect plans where one subgoal could clobber another. The plan was then simulated to see if it worked. Failures were then discovered through a comparison of the system's intentions and expectations with the actual result. Next, bug fixes were located which could be applied to the situation at hand. In this way, HACKER incrementally produced better and better plans until the desired criterion was met.

HACKER, like many of the early machine learning systems, was bound tightly to a specific domain. Not only were the different features of the system not all found in the same implementation, but the complex and specific nature of the knowledge libraries made domain independence a tough goal to achieve. Furthermore, HACKER was also making implicit assumptions about its world which actually affect the overall approach used. Since the system had ideal information, it was naturally possible to identify the precise cause of the failure. Several other researchers have done work in incrementally refining from failure. Among these are Chien and Gupta.

Chien's approach [Chien87] assumes a complete but intractable domain theory. The system operates in a machine–shop scheduling domain. The explanation of observed examples is made tractable through the use of persistence assumptions about world states. When the plan is executed, if failures occur, an explanation for the failure, ultimately due to an incorrect persistence assumption, is constructed and generalized for use in constructing a censor against using the plan in similar bad situations in the future. The system can additionally augment plans through understanding gained from observing unexpected situations where an observed plan succeeds despite censors which indicate it should not. Chien's system assumes it has perfect knowledge about the world but that, due to intractability, it must use assumptions in explanation construction. The GRASPER system must make use of approximations at a lower level to handle the problem of imperfect world knowledge. GRASPER has the capability to rate plausible failures and gives preference to recognizing plans which can deal with the largest set of

plausible failures. While Chien's system works by adding censors and attaching additional causal explanations to failing plans, GRASPER is able to construct a new plan incorporating more of the subtleties of the training example through tuning its approximations and regenerating the general plan. Furthermore, GRASPER may decide that some particular failure is tolerable according to its notion of operationality and may choose, for instance, to repeat the plan to see if it succeeds on a retry. GRASPER relies primarily on the uncertainty tolerant plans it develops to prevent errors in carrying out the plans in the real world. The approximations are a method for supporting this.

Gupta has also implemented a system which incrementally refines from failures [Gupta87]. The system consists of a planning system, called TRAP, which simulates plans it produces, generalizes failure explanations with the EBG algorithm, and adds additional constraints to the plan which serve to prevent such failures. In other words, it produces over-general plans which are then constrained using the incremental technique as failures are encountered.

Gupta's system is somewhat limited in that, although it may have a complete set of knowledge, it doesn't learn with an understander -- only from its own problem solver. One of the helpful aspects of using an understander in intractable domains is that you can use the observed task execution of others as a vital clue into how the system's knowledge should be accessed and applied. Furthermore, repeatedly constraining a plan, is not always desirable. Use of methods which can result in modification of

the structure of the plan is desirable, such as GRASPER's plan regeneration and Chien's use of learning from unexpected successes.

Minton's PRODIGY system has the capability of learning control knowledge from failures [Minton87]. A naive planner is used to construct a plan for achieving some goal. The planner makes implicit assumptions, for example, about the independence of subgoals. After a proof structure for achieving the goal has been constructed (or while it is being constructed for in–trial learning), observed failures are generalized into control rules which prevent the system from making unwise choices during such a proof. Rather than learning approximate rules, the system learns methods by which search can be made more efficient. This can be very useful but the system cannot effectively deal with intractability unless it has the ability to make and assess explicit approximations. Without such a method, PRODIGY can easily be overwhelmed with the size of search space.

Hammond's CHEF system [Hammond86] uses a case–based planner to learn from previous failures. When the system encounters a situation in which the plan it developed fails, it indexes the failure under a generalized set of features which indicate why the failure occurred as well as a set of features which help to predict the failure. During future planning, the system uses the failure predictive features to focus on avoiding similar failures during the construction of the new plan.

Central to the CHEF system is the notion of a powerful case–based planner that can select relevant failures and incorporate fixes for them into the current plan. Fur-

thermore, the possible plan fixes have to be selected from a fixed set already coded into the system. In contrast, the GRASPER system uses an understander to recognize and employ fixes for use in developing its own plan. For a planner to develop a complicated error tolerant plan by itself is frequently intractable. A good understanding mechanism is a much more effective way to solve the problem given that the domain is one where a good human problem solver is available to be observed.

7.4 Approximation and Learning

The author's earlier work used approximations in explanation generation to promote tractable EBL in mathematical domains [Bennett87]. This work motivated an investigation of how rules learned using approximations should be treated in the EBL framework and led to the current GRASPER system. The approximations used in the earlier system were of the common mathematical variety. Work is currently underway in demonstrating GRASPER in mathematical domains through defining common mathematical approximations within the GRASPER approximation framework.

Zweben's system simplifies control rules by eliminating some of their antecedents [Zweben88]. The antecedents are taken pairwise and their conditional probabilities examined over a set of instances where the rule was used. If one of the antecedents "almost always" evaluates to the same value as another, one of them can be dropped. A standard failure–based recovery approach is used if a control rule is results in bad system performance. Simplifications are then reconsidered if used in the failing control rule. Similarly Keller's METALEX system replaces a rule antecedent with TRUE if the

antecedent evaluates to TRUE in all instances seen and substitutes FALSE for an antecedent which evaluates to FALSE in all instances seen [Keller87b].

The systems of Zweben and Keller address the question of "when to approximate." Both seek to minimize the chance of the approximations being invalid through use of a number of observed training instances. Ultimately, a recovery scheme must be in place to remedy bad approximations. If several different approximations were possible, the process of deciding "what to approximate with" is added and makes the process even less tractable. When GRASPER is used in the robotics domain, the approximations are defined immediately for any objects in the world about which uncertainty exists. Since no object can be described precisely, approximations are mandatory. Naturally, it is open to question how precise the initial approximations are to be. Therefore, useful results from the issue of "when to approximate and with what" could be applied to GRASPER as well. One future issue is that the GRASPER system may have several good approximate representations possible for an object and one must be chosen.

Mostow and Fawcett highlight the importance of seeking the best theory through a search of an approximate theory space using the scope, accuracy, cost, and operational requirements of the theories [Mostow87]. These ideas are very much in line with the generality, accuracy, and efficiency aspects of operationality th we propose as a major guiding factor in the GRASPER approximation architecture. Mostow and Fawcett's current system, HDF, demonstrates how approximations from a specified set can be

applied manually to improve the scope, accuracy, cost, and operational requirements of a routine to calculate the cost of a search.

Ellman's POLLYANNA system uses approximations to a program playing the game of "hearts" [Ellman88]. It also uses a search through the approximate theory space to arrive at the set of approximations to be used. The search starts with the simplest theory and resorts only to the next more complicated theory when the simpler theory contradicts training examples. A "simpler" theory is defined as one having a set of assumptions which logically imply those of the less simple theory.

Doyle has constructed a system for refining causal explanations of how various devices operate [Doyle86]. His emphasis is on dealing with inconsistent theories: those ones which have abstracted away details of otherwise complete theories. Explanations produced with such theories can be incorrect. Doyle's approach is to make the theory less abstract to deal with encountered failures. The technique is only applied to learning causal descriptions of mechanisms. However, the major problem with this approach is the required predefined abstraction hierarchy. Moving to various levels of abstraction is a traversal of this hierarchy. In GRASPER, we need no such predefined hierarchy.

7.5 Operationality

Operationality is integral to explanation–based learning; every EBL system must have some method of handling it. Until recently, however, little effort had been made to formalize it. Other recently proposed definitions include those of Dietterich & Bennett, Keller, Minton, and Segre.

An often recognized paradox regarding operationality is: "If the goal was initially nonoperational to the system, how could the system ever make it operational?" In other words, in order to operationalize it, the system has to have some way of achieving the goal.

One remedy for the paradox, proposed by Dietterich and Bennett, is to create two distinct agents [Dietterich88]. One meta–agent's duty is to operationalize things and the other agent simply achieves goals using existing rules. Therefore, by this definition, initially the goal is operational for the meta–agent but nonoperational for the regular agent. Afterwards, the meta–agent has succeeded in making the goal operational for the regular agent through a change in the rules which the regular agent uses.

Such a multiple agent solution seems unnecessary. First, if the definition of operationality were continuous rather than binary, the situation is one in which the system makes a goal "more" operational. This by itself resolves the paradox. Furthermore, for systems that learn from an analysis of an externally produced solution rather than from an internal trace, the analysis is almost always easier. Therefore, for such understanding systems the paradox can be resolved as well. It was operational to understand initially, but nonoperational in a planning sense.

Keller proposes a definition of operationality based on *useability* and *utility* [Keller87a]. This definition focuses on systems concerned with concept recognition and the target for operationalization is therefore the *concept description*. A concept descrip-

tion is *useable* if it is evaluable by the performance system. However, in cases where the concept description is constructed by the system and has been shown to be valid, useability follows automatically as validity could have never been proven had it not been evaluable by the system.† Describing Keller's *useable* as *evaluable by the performance system* is not much of a clarification of the term and leaves a large part of operationality very vague. Keller does contribute a more precise definition for *utility*. This is a measure of to what degree the concept description meets the system's performance objectives. Since each system has its own performance objectives, this naturally means each system will measure utility in a different way. Keller's MetaLEX system uses an empirical approach for measuring utility [Keller87c]. MetaLEX's utility is defined as "achiev[ing] an X% improvement without a deterioration in effectiveness."

Minton's PRODIGY system learns search control rules to improve system performance [Minton88]. In order to decide which control rules are actually beneficial to the system, their utility is measured. In its simplest form, the utility of a rule is the difference between the average solution time without the rule and the average solution time with the rule. Minton's utility therefore corresponds to the efficiency aspects of our operationality definition.

Segre presents a definition which recognizes how real-world situations affect operationality. He has a five-fold definition including *obviousness*, *efficiency*, *generality*,

† Naturally, this does not hold for the multiple agent view where both agents may not have the same vocabulary of terms.

robustness, and *recoverability* [Segre88]. One plan is more obvious than another when it has a lower planning cost. Efficiency implies lower execution cost which has been defined in our model as plan economy. To be more general means for one plan to have broader applicability than another. This corresponds to the combination of precondition generality and effect generality. Robustness means "more likely to be successful in the real world." We have refined this to include aspects such as result accuracy, uncertainty tolerance and uncertainty reduction. Recoverability indicates a preference for plans whose failure modes are less extreme. This amounts to a combination of our probability of success and some account of the environment the system operates in. Overall, Segre's definition recognizes many of the same factors as our own but many of the terms could have been broken down to be more concrete. Segre's *robustness* is vague but constitutes an important part of the definition for use with real–world systems.

Braverman's IMEX system concerns itself with finding the most general way of expressing a rule while at the same time maintaining its operationality [Braverman88]. However, since generality is included as an aspect to our continuous definition of operationality, in our model this means searching for the most operational way in which a concept can be expressed. The search should really be directed using all the aspects of our operationality definition, not just on generality. However, if generality were the most important factor in operationality according to the user–designed weighting, a search keying on generality would be a good approach.

Hirsh points out that sometimes things are conditionally operational and presents a system called ROE for incorporating the conditions of operationality into the operational rule produced [Hirsh88]. This technique is useful where an explicit theory of operationality exists and can be reasoned about by the system. We plan to construct an explicit operationality theory for use in the next version of GRASPER.

8 FUTURE DIRECTIONS

There are many aspects of this system we plan to extend and adjunct areas that will be investigated. The ultimate goal of this work is to demonstrate how techniques embodied in the GRASPER system can be extended to the control of an actual robot arm using current sensing technology in a less than ideal environment.

The current GRASPER system functions with uncertain data about world objects. The system used a disembodied gripper which is assumed accurate. While it is true that uncertainty tolerant plans developed with this model can usually compensate for uncertainties in the arm as well, in order to be able to correct for all possible failures, the arm should be modeled as approximate as well. This extension primarily means the addition of domain knowledge which gives a model of how the robot arms functions. As with the initial approximation that objects were in the positions the approximate vision system indicated, the initial approximation for the robot arm would be that it should function just as textbook kinematic equations dictate. In reality, a number of complex problems plague this simple model. Robot arms, in general, have problems with repeatability: reproducing the same movement from the same commands. Eventually through observation, GRASPER would be able to learn plans that can deal with problems such as repeatability.

The current system uses a two-dimensional world to simplify some of the reasoning involved. Despite this simplification, much of the reasoning involved in determining the spatial relationships between objects is quite time consuming. One method we are

working on to address this problem and to make possible the move to a three–dimensional space is an approach to balancing accuracy and efficiency of these tests. The approximations being used in the current version of GRASPER are designed to simplify reasoning about uncertainties present in the world. Approximations could also be used to reduce the complexity of object representations, yielding greater efficiency for many of the spatial relationship tests.

Another way to increase the efficiency of the system would be to add in–trial learning. The issue faced when adding this capability is knowing what to learn. Learning every possible operator nullifies any advantage in efficiency that an in–trial learner might hope to gain. One solution is to use situations in which failures have occurred to learn control knowledge so that fruitless paths are not taken in the future [Minton87]. Another, is to learn rules for segments of the proof which were nontrivial in some way to discover [Minton85]. Usually, once an explanation was constructed, only the entire explanation formed the rule (unless in–trial learning had already taken place). It is useful to analyze the explanation in greater depth and to recognize repeated elements for use as learned rules. Such substructure discovery algorithms have already been implemented in similarity/difference–based learning (SDBL) for the purpose of generating new features for use in constructive induction and/or for reducing the complexity of certain examples [Holder88, Whitehall87].

GRASPER's current plans are somewhat limited by an imposed linear ordering on the actions. For example, in observing a human operator performing a grasp, it may

happen to observe the gripper's fingers being opened for the approach before the gripper is even in position for the approach. The general rule would indicate that this procedure is performed first. Such a rule is clearly over–specific as the finger adjustment could have been performed at any point prior to the final translate to the object to be grasped. What is needed is a plan which does not impose such a restrictive ordering on the actions. Work begun in this area includes that of Mooney [Mooney88], who presents an algorithm which extends the EGGS technique in the area of order generalization. Chien is also addressing this problem [Chien88].

Another interesting area involves limitations placed on generalization. If GRASPER were to observe a robot stacking a series of blocks, one above the other, in the real world, what generalized plan should be acquired? That is, how many blocks is an acquired plan good for stacking? Clearly, due to the inherent uncertainty about the positions, orientations, and shapes of the blocks, some limit exists. That limit is closely related to the uncertainty tolerance of the block stacking operation as acquired from the example. GRASPER should be able to make use of the determined uncertainty tolerance of the component operation of the plan to determine limitations on the overall plan.

GRASPER has been so tailored for working in real–world environments that we must not forget that some domains, such as mathematics, support precise formalisms. Here, approximations are all of the internal variety, capable of being revised by reasoning alone, not through interaction with some outside world. In mathematics, a check

can be used to see if approximate equations are consistent within some margin of error. If not, the approximations would have to be revised or eliminated to satisfy the goal of producing an efficient and reasonably accurate solution. We plan to use GRASPER in "perfect" domains such as mathematics through use of mathematical approximations in GRASPER's approximation framework.

The most significant area of future work for the GRASPER system is to incorporate explicitly the theory of operationality into the system. This constitutes a major step forward in system flexibility. It would allow GRASPER to function in many environments simply by changing the weighting of the different aspects of operationality. One can even envision algorithms that may allow the system to change the weighting gradually itself as it performs more work in a certain environment in order to tailor itself to it.

The major goals for explicit operationality involve developing a logic for expression of operationality measures, developing good approximate measures of the different aspects of operationality with respect to a plan, and illustrating techniques in system design which are favorable under our proposed definition of operationality.

9 CONCLUSIONS

This thesis demonstrates a powerful technique for using approximations to represent information about uncertain objects. Approximation allows a system to reason economically about world objects and handle their possible uncertainties. These are key factors in using a system in complex real–world environments.

Explicit reasoning about approximations is carried out only when planning concepts are acquired or when failures are diagnosed. The plans themselves do no explicit reasoning about uncertainties. This promotes much more economical problem solving by the system. Failures which do occur due to an approximation–based plan are corrected by the system through revision of participating planning concepts. When a plan fails due to bad approximations of uncertain objects, the approximations are refined and a new planning concept acquired with sufficient uncertainty tolerance to prevent the failure.

The entire system is moderated by a complete definition of operationality. The nine aspect definition deals with generality, economy, and uncertainty. The definition helps us to see the important tradeoffs that approximation permits. It demonstrates how important uncertainty tolerance is for real–world systems. The definition is used in the understander to make decisions regarding the elimination or modification of actions not supported by the approximate model. It is also used in the failure recovery mechanism to favor one revised plan over another. This approach to learning and using uncertainty

tolerant plans through approximations offers great promise for use in complex real–world environments.

APPENDIX 1 FORMING THE GENERAL RULE

Once an explanation has been produced, in order for the system to take advantage of the new knowledge, a general rule or rules must be produced from it. GRASPER will produce a single general rule for achieving a goal using an explanation of the specific way in which that goal was achieved.

The explanation, as described in Chapter 5, has a trace of the rules which are used and how they are connected. In order for an antecedent of one rule to be supported by the consequent of another requires their corresponding patterns to be equal. The first step involves producing a list of variable/value pairs obtained through unification of these patterns. This is the first phase of the EGGS generalization algorithm and is stated formally below [Mooney86]:

let γ be the null substitution {}
for each equality between patterns p_i and p_j in the explanation structure do
    let θ be the MGU of $p_i\ \gamma$ and $p_j\ \gamma$
    let γ be $\gamma\theta$

Next, a rule is formed whose consequents are the consequents of applying all the rules as specified in the explanation structure. The rule's antecedents are the leaves of the explanation structure. The rule also specifies the actions to be carried out if there were any specified in the explanation.

The substitution list γ produced in the first step is now used to instantiate the rule produced in the previous step. This is accomplished as follows:

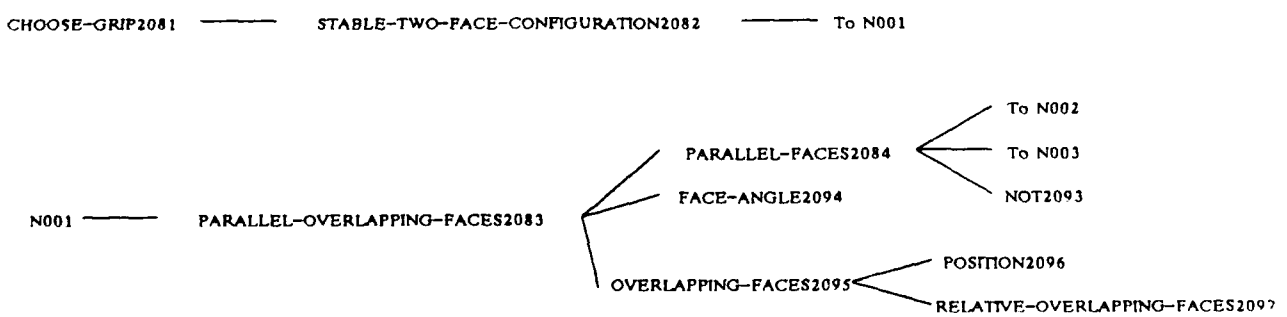
for each pattern p_k in the explanation structure do

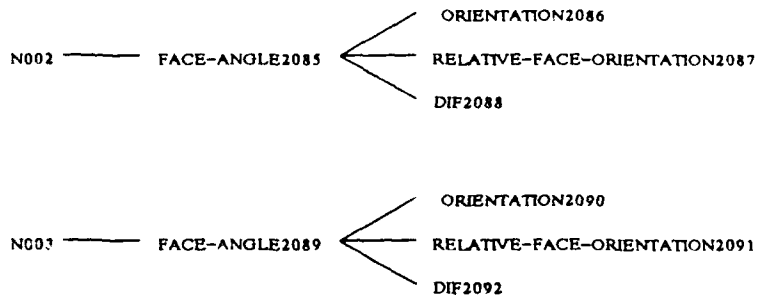
replace p_k with $p_k \gamma$

To illustrate how the algorithm is applied, consider the procedure for choosing a grip on an object. Alternatively, this can be viewed as verifying that a grip is a correct one. Suppose our goal is to choose a grip for square2 as specified by

```
choose-grip
  gripper1 <---------------------------------------- Gripper Name
  square2 <----------------------------------------- Object
  ?target-x <--------------------------------------- Target X-Coordinate
  ?target-y <--------------------------------------- Target Y-Coordinate
  ?target-angle <----------------------------------- Target Gripper Angle
  ?target-width <----------------------------------- Target Gripper Width
  ?face1 <------------------------------------------ One Face Of The Object
  ?face2 <------------------------------------------ Another Face Of The Object
  ?max-position-error <----------------------------- Maximum Position Error
  ?max-orientation-error <-------------------------- Maximum Orientation Error
```

The planner or understander constructs the explanation for the goal. The explanation appears below in its specific form (side effects excluded):





CHOOSE-GRIP2081
(CHOOSE-GRIP GRIPPER1 SQUARE2 4.5 3.5 0.0 3.0 F1 F3 0 0)
DIF2088 DIF2092
(DIF -0.0 0.0 0)
FACE-ANGLE2085 FACE-ANGLE2094
(FACE-ANGLE SQUARE2 F1 0.0 0 0)
FACE-ANGLE2089
(FACE-ANGLE SQUARE2 F3 0.0 0 0)
NOT2093
(NOT (EQUAL F1 F3))
ORIENTATION2086 ORIENTATION2090
(ORIENTATION SQUARE2 0 0)
OVERLAPPING-FACES2095
(OVERLAPPING-FACES SQUARE2 F1 F3 4.5 3.5 3.0 0 0)
PARALLEL-FACES2084
(PARALLEL-FACES SQUARE2 F1 F3 0)
PARALLEL-OVERLAPPING-FACES2083
(PARALLEL-OVERLAPPING-FACES SQUARE2 4.5 3.5 0.0 3.0 F1 F3 0 0)
POSITION2096
(POSITION SQUARE2 4.5 3.5 0)
RELATIVE-FACE-ORIENTATION2087
(RELATIVE-FACE-ORIENTATION SQUARE2 F1 -0.0 0)
RELATIVE-FACE-ORIENTATION2091
(RELATIVE-FACE-ORIENTATION SQUARE2 F3 -0.0 0)
RELATIVE-OVERLAPPING-FACES2097
(RELATIVE-OVERLAPPING-FACES SQUARE2 F1 F3 0.0 0.0 3.0 0)
STABLE-TWO-FACE-CONFIGURATION2082
(STABLE-TWO-FACE-CONFIGURATION SQUARE2 4.5 3.5 0.0 3.0 F1 F3 0 0)

The following rules and facts are used in one explanation for the above *choose-grip* operator. The rules shown are uniquified versions of those from the knowledge-base.

```
(rule :cons
      (choose-grip ?gripper2023
                   ?object2024
                   ?target-x2025
                   ?target-y2026
                   ?target-angle2027
                   ?target-width2028
                   ?face12029
                   ?face22030
                   ?perror2031
                   ?oerror2032)
      :ants
      (stable-two-face-configuration ?object2024
                                     ?target-x2025
                                     ?target-y2026
                                     ?target-angle2027
                                     ?target-width2028
                                     ?face12029
                                     ?face22030
                                     ?perror2031
                                     ?oerror2032))
```

R1

```
(rule :cons
      (stable-two-face-configuration ?object2033
                                     ?target-x2034
                                     ?target-y2035
                                     ?target-angle2036
                                     ?target-width2037
                                     ?face12038
                                     ?face22039
                                     ?perror2040
                                     ?oerror2041)
      :ants
      (parallel-overlapping-faces ?object2033
                                  ?target-x2034
                                  ?target-y2035
                                  ?target-angle2036
                                  ?target-width2037
                                  ?face12038
                                  ?face22039
                                  ?perror2040
                                  ?oerror2041))
```

R2

```
(rule :cons
      (parallel-overlapping-faces ?object2042
                                  ?target-x2043
                                  ?target-y2044
                                  ?target-angle2045
                                  ?target-width2046
                                  ?face12047
                                  ?face22048
                                  ?perror2049
                                  ?oerror2050)
      :ants
      (parallel-faces ?object2042 ?face12047 ?face22048 ?oerror2050)
      (face-angle ?object2042 ?face12047 ?target-angle2045
      (overlapping-faces ?object2042          ?oerror2050 0)
                         ?face12047
                         ?face22048
                         ?target-x2043
                         ?target-y2044
                         ?target-width2046
                         ?perror2049
                         ?serror2051))
```

R3

```
(rule :cons
      (parallel-faces ?object2052
                      ?face12053
                      ?face22054
                      ?oerror2055)
      :ants
      (face-angle ?object2052 ?face12053 ?angle2056 ?oerror2055 0)
      (face-angle ?object2052 ?face22054 ?angle2056 ?oerror2055 0)
      (not (equal ?face12053 ?face22054)))
```

R4

```
(rule :cons
      (face-angle ?object2057
                  ?face2058
                  ?face-angle2059
                  ?oerror2060
                  ?rfe-error2061)
      :ants
      (orientation ?object2057 ?angle2062 ?oerror2060)
      (relative-face-orientation ?object2057
                                 ?face2058
                                 ?relative-face-angle2063
                                 ?rfe-error2061)
      (dif ?relative-face-angle2063 ?face-angle2059 ?angle2062))
```

R5

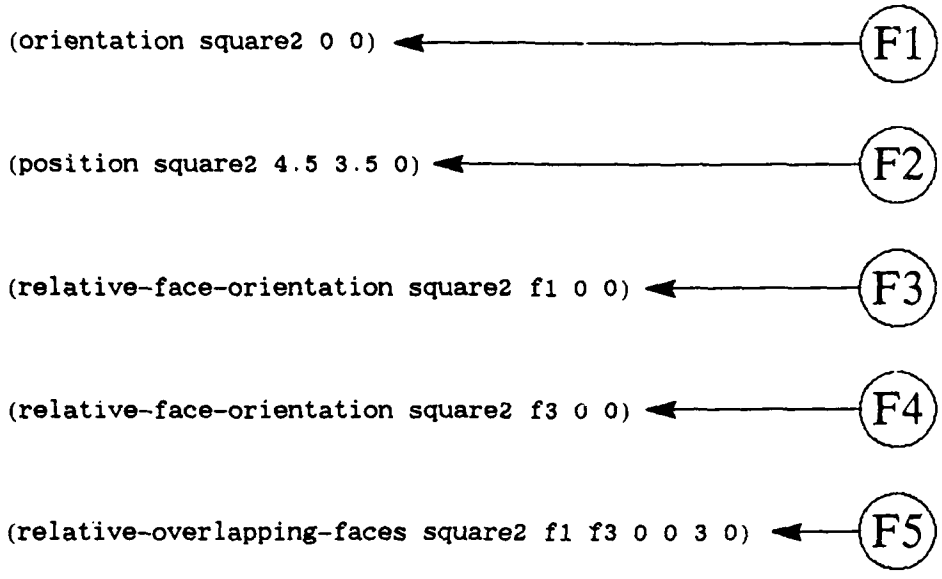
```
(rule :cons
      (face-angle ?object2065
                  ?face2066
                  ?face-angle2067
                  ?oerror2068
                  ?rfe-error2069)
      :ants
      (orientation ?object2065 ?angle2070 ?oerror2068)
      (relative-face-orientation ?object2065
                                 ?face2066
                                 ?relative-face-angle2071
                                 ?rfe-error2069)
      (dif ?relative-face-angle2071 ?face-angle2067 ?angle2070))
```

R6

```
(rule :cons
      (overlapping-faces ?object2073
                         ?face12074
                         ?face22075
                         ?x2076
                         ?y2077
                         ?separation2078
                         ?perror2079
                         ?of-error2080)
      :ants
      (position ?object2073 ?x2076 ?y2077 ?perror2079)
      (relative-overlapping-faces ?object2073
                                  ?face12074
                                  ?face22075
                                  0
                                  0
                                  ?separation2078
                                  ?of-error2080))
```

R7

Facts used in the explanation:



Facts and built–in predicates are considered operational. That is, they will form the antecedents of our general rule. It is also possible to explicitly mark rules in the database as operational should this be desirable. Using the specific/general binding list approach from the EGGS algorithm, the explanation predicates are unified. Unifications between the goal predicate *choose–grip* and the consequent of the *choose–grip* rule as well as all unifications to operational predicates do not appear in the general bindings as they are particular to the current example. All other unifications appear on both the general and specific lists. The specific and general lists for the above example are shown below:

SPECIFIC-BINDING-LIST

```
(T (?GRIPPER2023 GRIPPER1)
   (?OBJECT2024 SQUARE2)
   (?TARGET-X ?TARGET-X2025)
   (?TARGET-Y ?TARGET-Y2026)
   (?TARGET-ANGLE ?TARGET-ANGLE2027)
   (?TARGET-WIDTH ?TARGET-WIDTH2028)
   (?FACE1 ?FACE12029)
   (?FACE2 ?FACE22030)
   (?MAX-POSITION-ERROR ?PERROR2031)
   (?MAX-ORIENTATION-ERROR ?OERROR2032)        Goal & R1
   (?OBJECT2033 SQUARE2)
   (?TARGET-X2025 ?TARGET-X2034)
   (?TARGET-Y2026 ?TARGET-Y2035)
   (?TARGET-ANGLE2027 ?TARGET-ANGLE2036)
   (?TARGET-WIDTH2028 ?TARGET-WIDTH2037)       R1 & R2
   (?FACE12029 ?FACE12038)
   (?FACE22030 ?FACE22039)
   (?PERROR2031 ?PERROR2040)
   (?OERROR2032 ?OERROR2041)
   (?OBJECT2042 SQUARE2)
   (?TARGET-X2034 ?TARGET-X2043)
   (?TARGET-Y2035 ?TARGET-Y2044)
   (?TARGET-ANGLE2036 ?TARGET-ANGLE2045)
   (?TARGET-WIDTH2037 ?TARGET-WIDTH2046)       R2 & R3
   (?FACE12038 ?FACE12047)
   (?FACE22039 ?FACE22048)
   (?PERROR2040 ?PERROR2049)
   (?OERROR2041 ?OERROR2050)
   (?OBJECT2052 SQUARE2)
   (?FACE12047 ?FACE12053)
   (?FACE22048 ?FACE22054)                     R3 & R4
   (?OERROR2050 ?OERROR2055)
   (?OBJECT2057 SQUARE2)
   (?FACE12053 ?FACE2058)
   (?ANGLE2056 ?FACE-ANGLE2059)                R4 & R5
   (?OERROR2055 ?OERROR2060)
   (?RFE-ERROR2061 0)
   (?ANGLE2062 0)                              R5 & F1
   (?OERROR2060 0)
   (?FACE2058 F1)                              R5 & F3
   (?RELATIVE-FACE-ANGLE2063 -0.0)
   (?FACE-ANGLE2059 0.0)                       R5 & DIF Procedure
   (?OBJECT2065 SQUARE2)
   (?FACE22054 ?FACE2066)
   (?FACE-ANGLE2067 0.0)                       R4 & R6
   (?OERROR2068 0)
   (?RFE-ERROR2069 0)
   (?ANGLE2070 0)                              R6 & F1
   (?FACE2066 F3)                              R6 & F4
   (?RELATIVE-FACE-ANGLE2071 -0.0)
```

SPECIFIC-BINDING-LIST (continued)

```
(?TARGET-ANGLE2045 ?FACE-ANGLE2059)                    R3 & R5
(?OBJECT2073 SQUARE2)
(?FACE12074 F1)
(?FACE22075 F3)
(?TARGET-X2043 ?X2076)                                 R3 & R7
(?TARGET-Y2044 ?Y2077)
(?TARGET-WIDTH2046 ?SEPARATION2078)
(?PERROR2049 ?PERROR2079)
(?SERROR2051 ?OF-ERROR2080)
(?X2076 4.5)                                           R7 & F2
(?Y2077 3.5)
(?PERROR2079 0)
(?SEPARATION2078 3.0)                                  R7 & F5
(?OF-ERROR2080 0))
```

GENERAL-BINDING-LIST

```
(T (?OBJECT2024 ?OBJECT2033)                          ┐
   (?TARGET-X2025 ?TARGET-X2034)                      │
   (?TARGET-Y2026 ?TARGET-Y2035)                      │
   (?TARGET-ANGLE2027 ?TARGET-ANGLE2036)              │
   (?TARGET-WIDTH2028 ?TARGET-WIDTH2037)              │ R1 & R2
   (?FACE12029 ?FACE12038)                            │
   (?FACE22030 ?FACE22039)                            │
   (?PERROR2031 ?PERROR2040)                          │
   (?OERROR2032 ?OERROR2041)                          ┘
   (?OBJECT2033 ?OBJECT2042)                          ┐
   (?TARGET-X2034 ?TARGET-X2043)                      │
   (?TARGET-Y2035 ?TARGET-Y2044)                      │
   (?TARGET-ANGLE2036 ?TARGET-ANGLE2045)              │
   (?TARGET-WIDTH2037 ?TARGET-WIDTH2046)              │ R2 & R3
   (?FACE12038 ?FACE12047)                            │
   (?FACE22039 ?FACE22048)                            │
   (?PERROR2040 ?PERROR2049)                          │
   (?OERROR2041 ?OERROR2050)                          ┘
   (?OBJECT2042 ?OBJECT2052)                          ┐
   (?FACE12047 ?FACE12053)                            │ R3 & R4
   (?FACE22048 ?FACE22054)                            │
   (?OERROR2050 ?OERROR2055)                          ┘
   (?OBJECT2052 ?OBJECT2057)                          ┐
   (?FACE12053 ?FACE2058)                             │
   (?ANGLE2056 ?FACE-ANGLE2059)                       │ R4 & R5
   (?OERROR2055 ?OERROR2060)                          │
   (?RFE-ERROR2061 0)                                 ┘
   (?OBJECT2057 ?OBJECT2065)                          ┐
   (?FACE22054 ?FACE2066)                             │
   (?FACE-ANGLE2059 ?FACE-ANGLE2067)                  │ R4 & R6
   (?OERROR2060 ?OERROR2068)                          │
   (?RFE-ERROR2069 0)                                 ┘
   (?TARGET-ANGLE2045 ?FACE-ANGLE2059)                   R3 & R5
   (?OBJECT2065 ?OBJECT2073)                          ┐
   (?FACE2058 ?FACE12074)                             │
   (?FACE2066 ?FACE22075)                             │
   (?TARGET-X2043 ?X2076)                             │ R3 & R7
   (?TARGET-Y2044 ?Y2077)                             │
   (?TARGET-WIDTH2046 ?SEPARATION2078)                │
   (?PERROR2049 ?PERROR2079)                          │
   (?SERROR2051 ?OF-ERROR2080))                       ┘
```

The resulting generalized rule for *choose-grip* appears as follows:

```
(rule :cons
      (choose-grip ?gripper2023
                   ?object2073
                   ?x2076
                   ?y2077
                   ?face-angle2067
                   ?separation2078
                   ?face12074
                   ?face22075
                   ?perror2079
                   ?oerror2068)
      :ants
      (orientation ?object2073 ?angle2062 ?oerror2068)
      (relative-face-orientation ?object2073
                                 ?face12074
                                 ?relative-face-angle2063
                                 0)
      (dif ?relative-face-angle2063 ?face-angle2067 ?angle2062)
      (orientation ?object2073 ?angle2070 ?oerror2068)
      (relative-face-orientation ?object2073
                                 ?face22075
                                 ?relative-face-angle2071
                                 0)
      (dif ?relative-face-angle2071 ?face-angle2067 ?angle2070)
      (not (equal ?face12074 ?face22075))
      (position ?object2073 ?x2076 ?y2077 ?perror2079)
      (relative-overlapping-faces ?object2073
                                  ?face12074
                                  ?face22075
                                  0
                                  0
                                  ?separation2078
                                  ?of-error2080))
```

APPENDIX 2 A COMPLETE SAMPLE RUN

```
> (demo)
Reloading Rules & Object Hierarchy...
;;; Loading source file "rules.lisp"
Observation Started...
Action Observed...

OBSERVED-APPLY-FORCE
  GRIPPER1 <---------- Gripper Name
  5 <---------- Motor Force Applied
  0 <---------- Numeric Starting Time
  1 <---------- Numeric Ending Time
  -7.5 <---------- Current Gripper X-Coordinate
  8.5 <---------- Current Gripper Y-Coordinate
  -9.5 <---------- Target X-Coordinate
  3.5 <---------- Target Y-Coordinate
  4 <---------- Gripper Beam Length
  2 <---------- Gripper Beam Width
  3 <---------- Gripper Finger Length
  2 <---------- Gripper Finger Width
  2 <---------- Current Gripper Finger Separation
  -90 <---------- Current Gripper Angle

Action Observed...

OBSERVED-APPLY-TORQUE
  GRIPPER1 <---------- Gripper Name
  5 <---------- Motor Force Applied
  1 <---------- Numeric Starting Time
  2 <---------- Numeric Ending Time
  -90 <---------- Current Gripper Angle
  0 <---------- Target Gripper Angle
  4 <---------- Gripper Beam Length
  2 <---------- Gripper Beam Width
  3 <---------- Gripper Finger Length
  2 <---------- Gripper Finger Width
  2 <---------- Current Gripper Finger Separation
  -9.5 <---------- Current Gripper X-Coordinate
  3.5 <---------- Current Gripper Y-Coordinate

Action Observed...

OBSERVED-OPEN-FINGERS
  GRIPPER1 <---------- Gripper Name
  5 <---------- Motor Force Applied
  2 <---------- Numeric Starting Time
  3 <---------- Numeric Ending Time
  2 <---------- Current Gripper Finger Separation
```

3.75 <---------- Target Gripper Width
-9.5 <---------- Current Gripper X-Coordinate
3.5 <---------- Current Gripper Y-Coordinate
0 <---------- Current Gripper Angle
2 <---------- Current Gripper Finger Separation
2 <---------- Gripper Beam Width
4 <---------- Gripper Beam Length
2 <---------- Gripper Finger Width
3 <---------- Gripper Finger Length

Action Observed...

OBSERVED-APPLY-FORCE
GRIPPER1 <---------- Gripper Name
5 <---------- Motor Force Applied
3 <---------- Numeric Starting Time
4 <---------- Numeric Ending Time
-9.5 <---------- Current Gripper X-Coordinate
3.5 <---------- Current Gripper Y-Coordinate
4.5 <---------- Target X-Coordinate
3.5 <---------- Target Y-Coordinate
4 <---------- Gripper Beam Length
2 <---------- Gripper Beam Width
3 <---------- Gripper Finger Length
2 <---------- Gripper Finger Width
3.75 <---------- Current Gripper Finger Separation
0 <---------- Current Gripper Angle

Action Observed...

OBSERVED-CLOSE-FINGERS
GRIPPER1 <---------- Gripper Name
5 <---------- Motor Force Applied
4 <---------- Numeric Starting Time
5 <---------- Numeric Ending Time
3.75 <---------- Current Gripper Finger Separation
3 <---------- Target Gripper Width
4.5 <---------- Current Gripper X-Coordinate
3.5 <---------- Current Gripper Y-Coordinate
0 <---------- Current Gripper Angle
3.75 <---------- Current Gripper Finger Separation
2 <---------- Gripper Beam Width
4 <---------- Gripper Beam Length
2 <---------- Gripper Finger Width
3 <---------- Gripper Finger Length

Observation Ended...
Seeking To Understand Observations...
No Errors Queued, Performing No Focusing.

Creating suggestion list...

Queue: (OBSERVED-APPLY-FORCE OBSERVED-APPLY-TORQUE OB-
SERVED-OPEN-FINGERS OBSERVED-APPLY-FORCE OBSERVED-CLOSE-FINGERS)

Suggestions for OBSERVED-APPLY-FORCE are (APPLY-FORCE)...
Working on (OBSERVED-APPLY-FORCE GRIPPER1 5 0 1 -7.5 8.5 -9.5 3.5 4 2 3 2 2 -90)
which suggests (APPLY-FORCE)...Attempting Suggestion Link-Up...Retrieving Suggested
Rule...Found Corresponding Antecedent...Successful. Found
Corresponding Antecedent...Successful...Working top-down to prove suggestion...Working
top-down to prove suggestion...Explanation found...Queueing (APPLY-FORCE GRIPPER1 -2.0
-5.0 5 0 1 (GRIPPER-CLEAR-TRANSLATE GRIPPER1 -7.5
8.5 -9.5 3.5 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 2 -90) (AT-COORDINATES -9.5
3.5 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIP-
PER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Suggestions for OBSERVED-APPLY-TORQUE are (APPLY-TORQUE)...
Working on (OBSERVED-APPLY-TORQUE GRIPPER1 5 1 2 -90 0 4 2 3 2 2 -9.5 3.5)
which suggests (APPLY-TORQUE)...Attempting Suggestion Link-Up...Retrieving Suggested
Rule...Found Corresponding Antecedent...Successful...Working top
-down to prove suggestion...Explanation found...Queueing (APPLY-TORQUE GRIPPER1 90.0
5 1 2 (GRIPPER-CLEAR-ROTATE GRIPPER1 -90 0 (GRIPPER1 SQUARE1 SQUARE2) 4 2
3 2 2 -9.5 3.5) (AT-ANGLE 0 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5)
(GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPA-
RATION GRIPPER1 2)))...
Suggestions for OBSERVED-OPEN-FINGERS are (OPEN-FINGERS)...
Working on (OBSERVED-OPEN-FINGERS GRIPPER1 5 2 3 2 3.75 -9.5 3.5 0 2 2 4 2 3)
which suggests (OPEN-FINGERS)...Attempting Suggestion Link-Up...Retrieving Suggested
Rule...Found Corresponding Antecedent...Successful...Working
top-down to prove suggestion...Explanation found...Queueing (OPEN-FINGERS GRIPPER1
1.75 5 2 3 (GRIPPER-CLEAR-FINGER-ADJUST GRIPPER1 2 3.75 (GRIPPER1 SQUARE1
SQUARE2) -9.5 3.5 0 2 2 4 2 3) (AT-WIDTH 3.75 GRIPPER1) ((GRIPPER-X
GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIP-
PER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for OBSERVED-APPLY-FORCE are (APPLY-FORCE)...
Working on (OBSERVED-APPLY-FORCE GRIPPER1 5 3 4 -9.5 3.5 4.5 3.5 4 2 3 2 3.75 0)
which suggests (APPLY-FORCE)...Attempting Suggestion Link-Up...Retrieving Suggested
Rule...Found Corresponding Antecedent...Successful...Found
Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation
found...Queueing (APPLY-FORCE GRIPPER1 14.0 0.0 5 3 4 (GRIPPER-APPROACH-SE-
QUENCE GRIPPER1 14.0 0.0 SQUARE2 F1 F3 4.5 3.5 0 3.75 -9.5
3.5 ?OB-POLY733 ?GCX734 ?GCY735 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 0)
(AT-COORDINATES 4.5 3.5 GRIPPER1) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y
GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIP-
PER1 3.75))
)...
Suggestions for OBSERVED-CLOSE-FINGERS are (CLOSE-FINGERS)...
Working on (OBSERVED-CLOSE-FINGERS GRIPPER1 5 4 5 3.75 3 4.5 3.5 0 3.75 2 4 2 3)
which suggests (CLOSE-FINGERS)...Attempting Suggestion Link-Up...Retrieving Suggested
Rule...Found Corresponding Antecedent...Successful...Work

ing top-down to prove suggestion...Explanation found...Queueing (CLOSE-FINGERS GRIPPER1 -0.75 5 4 5 (GRIPPING-SEQUENCE GRIPPER1 SQUARE2 F1 F3 5 3.75 3) (GRIPPER-GRIP-PRESSURE GRIPPER1 5) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3)))...

Queue: (APPLY-FORCE APPLY-TORQUE OPEN-FINGERS APPLY-FORCE CLOSE-FINGERS)

Suggestions for APPLY-FORCE are (MOVE-GRIPPER-CLEAR APPROACH-TARGET-CLEAR)...
Working on (APPLY-FORCE GRIPPER1 -2.0 -5.0 5 0 1 (GRIPPER-CLEAR-TRANSLATE GRIPPER1 -7.5 8.5 -9.5 3.5 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 2 -90) (AT-COORDINATES -9.5 3.5 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))) which suggests (MOVE-GRIPPER-CLEAR APPROACH-TARGET-CLEAR)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecede
nt...Successful...Found Corresponding Antecedent...Unsuccessful...Working top-down to prove suggestion...Explanation found...Queueing (MOVE-GRIPPER-CLEAR GRIPPER1 -2.0 -5.0 -7.5 8.5 -90 2 -9.5 3.5 2 3 4 2 5 0 1 (GRIPPER1 SQU
ARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Unsuccessful...Suggestions for APPLY-TORQUE are (ROTATE-GRIPPER-CLEAR)...
Working on (APPLY-TORQUE GRIPPER1 90.0 5 1 2 (GRIPPER-CLEAR-ROTATE GRIPPER1 -90 0 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 2 -9.5 3.5) (AT-ANGLE 0 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPE
R1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))) which suggests (ROTATE-GRIPPER-CLEAR)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Found Corresponding Antecedent.
..Unsuccessful...Working top-down to prove suggestion...Explanation found...Queueing (ROTATE-GRIPPER-CLEAR GRIPPER1 90.0 -9.5 3.5 -90 2 0 2 3 4 2 5 1 2 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER
1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Suggestions for OPEN-FINGERS are (ADJUST-FINGERS)...
Working on (OPEN-FINGERS GRIPPER1 1.75 5 2 3 (GRIPPER-CLEAR-FINGER-ADJUST GRIPPER1 2 3.75 (GRIPPER1 SQUARE1 SQUARE2) -9.5 3.5 0 2 2 4 2 3) (AT-WIDTH 3.75 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75))) which suggests (ADJUST-FINGERS)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Found Corresponding A
ntecedent...Unsuccessful...Working top-down to prove suggestion...Explanation found...Queueing (ADJUST-FINGERS GRIPPER1 1.75 5 2 3 (GRIPPER-CLEAR-FINGER-ADJUST GRIPPER1 2 3.75 (GRIPPER1 SQUARE1 SQUARE2) -9.5 3.5 0 2 2 4 2 3)
(AT-WIDTH 3.75 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1

3.75)))...
Suggestions for APPLY-FORCE are (MOVE-GRIPPER-CLEAR APPROACH-TARGET-CLEAR)...
Working on (APPLY-FORCE GRIPPER1 14.0 0.0 5 3 4 (GRIPPER-APPROACH-SEQUENCE GRIPPER1 14.0 0.0 SQUARE2 F1 F3 4.5 3.5 0 3.75 -9.5 3.5 ?OB-POLY733 ?GCX734 ?GCY735 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 0) (AT-COORDINATES 4.5 3.5 GRIPPER1) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75))) which suggests (MOVE-GRIPPER-CLEAR APPROACH-TARGET-CLEAR)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Unsuccessful...Found Corresponding Antecedent...Unsuccessful...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful
...Working top-down to prove suggestion...Explanation found...Queueing (APPROACH-TARGET-CLEAR GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 2 3 4 2 5 3 4 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)) F1 F3)...
Suggestions for CLOSE-FINGERS are (GRIP ADJUST-FINGERS)...
Working on (CLOSE-FINGERS GRIPPER1 -0.75 5 4 5 (GRIPPING-SEQUENCE GRIPPER1 SQUARE2 F1 F3 5 3.75 3) (GRIPPER-GRIP-PRESSURE GRIPPER1 5) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3))) which suggests (GRIP ADJUST-FINGERS)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (GRIP GRIPPER1 5 SQUARE2 F1 F3 4 5 3.75 3)...
Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Unsuccessful...Found Corresponding Antecedent...Unsuccessful...

Queue: (MOVE-GRIPPER-CLEAR ROTATE-GRIPPER-CLEAR ADJUST-FINGERS APPROACH-TARGET-CLEAR GRIP)

Suggestions for MOVE-GRIPPER-CLEAR are (MOVE-FOR-APPROACH)...
Working on (MOVE-GRIPPER-CLEAR GRIPPER1 -2.0 -5.0 -7.5 8.5 -90 2 -9.5 3.5 2 3 4 2 5 0 1 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))) which suggests (MOVE-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (MOVE-FOR-APPROACH GRIPPER1 -7.5 8.5 -90 2 -9.5 3.5 2 3 4 2 5 0 1 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Suggestions for ROTATE-GRIPPER-CLEAR are (ROTATE-FOR-APPROACH)...
Working on (ROTATE-GRIPPER-CLEAR GRIPPER1 90.0 -9.5 3.5 -90 2 0 2 3 4 2 5 1 2 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))) which suggests (ROTATE-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (ROTATE-FO

R-APPROACH GRIPPER1 -9.5 3.5 -90 2 0 2 3 4 2 5 1 2 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Suggestions for ADJUST-FINGERS are (ADJUST-FINGERS-CLEAR)...
Working on (ADJUST-FINGERS GRIPPER1 1.75 5 2 3 (GRIPPER-CLEAR-FINGER-ADJUST GRIPPER1 2 3.75 (GRIPPER1 SQUARE1 SQUARE2) -9.5 3.5 0 2 2 4 2 3) (AT-WIDTH 3.75 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPP
ER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75))) which suggests (ADJUST-FINGERS-CLEAR)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Found Corresp
onding Antecedent...Unsuccessful...Working top-down to prove suggestion...Explanation found...Queueing (ADJUST-FINGERS-CLEAR GRIPPER1 1.75 ?GRIPPER-X1990 ?GRIPPER-Y1991 ?GRIPPER-ANGLE1992 2 3.75 2 3 4 2 5 2 3 (GRIPPER1 SQUAR
E1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for APPROACH-TARGET-CLEAR are (APPROACH-TARGET)...
Working on (APPROACH-TARGET-CLEAR GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 2 3 4 2 5 3 4 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRI
PPER1 3.75)) F1 F3) which suggests (APPROACH-TARGET)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queu
eing (APPROACH-TARGET GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 0 3.0 0 4 ?GRIPPING-FORCE2032 5 F1 F3 0 0)...
Suggestions for GRIP are (ACHIEVE-FRICTIONAL-FORCE)...
Working on (GRIP GRIPPER1 5 SQUARE2 F1 F3 4 5 3.75 3) which suggests (ACHIEVE-FRICTIONAL-FORCE)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (ACHIEVE-FRICTIONAL-FORCE GRIPPER1 SQUARE2 5 0 5 ?GRIPPING-FORCE2032 5)...

Queue: (MOVE-FOR-APPROACH ROTATE-FOR-APPROACH ADJUST-FINGERS-CLEAR APPROACH-TARGET ACHIEVE-FRICTIONAL-FORCE)

Suggestions for MOVE-FOR-APPROACH are (PREPARE-FOR-APPROACH)...
Working on (MOVE-FOR-APPROACH GRIPPER1 -7.5 8.5 -90 2 -9.5 3.5 2 3 4 2 5 0 1 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))) which suggests (PREPARE-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (PREPARE
-FOR-APPROACH GRIPPER1 -9.5 3.5 0 3.75 2 3 4 2 5 0 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for ROTATE-FOR-APPROACH are (PREPARE-FOR-APPROACH)...
Working on (ROTATE-FOR-APPROACH GRIPPER1 -9.5 3.5 -90 2 0 2 3 4 2 5 1 2 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5)

(GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))) which suggests (PREPARE-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (PREPARE-FOR-AP
PROACH GRIPPER1 -9.5 3.5 0 3.75 2 3 4 2 5 0 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for ADJUST-FINGERS-CLEAR are (ADJUST-FINGERS-FOR-APPROACH)...
Working on (ADJUST-FINGERS-CLEAR GRIPPER1 1.75 -9.5 3.5 0 2 3.75 2 3 4 2 5 2 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.7 5))) which suggests (ADJUST-FINGERS-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queuein
g (ADJUST-FINGERS-FOR-APPROACH GRIPPER1 -9.5 3.5 0 2 3.75 2 3 4 2 5 2 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...

Suggestions for APPROACH-TARGET are (ACHIEVE-FRICTIONAL-FORCE)...
Working on (APPROACH-TARGET GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 0 3.0 0 4 ?GRIPPING-FORCE2032 5 F1 F3 0 0) which suggests (ACHIEVE-FRICTIONAL-FORCE)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Co
rresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (ACHIEVE-FRICTIONAL-FORCE GRIPPER1 SQUARE2 5 0 5 ?GRIPPING-FORCE2032 5)...
Suggestions for ACHIEVE-FRICTIONAL-FORCE are (GRASP)...
Working on (ACHIEVE-FRICTIONAL-FORCE GRIPPER1 SQUARE2 5 0 5 ?GRIPPING-FORCE2032 5) which suggests (GRASP)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working to
p-down to prove suggestion...Explanation found...Queueing (GRASP GRIPPER1 SQUARE2 ?GRIPPING-FORCE2032 5 0 5)...

Queue: (PREPARE-FOR-APPROACH ADJUST-FINGERS-FOR-APPROACH ACHIEVE-FRICTIONAL-FORCE GRASP)

Suggestions for PREPARE-FOR-APPROACH are (APPROACH-TARGET)...
Working on (PREPARE-FOR-APPROACH GRIPPER1 -9.5 3.5 0 3.75 2 3 4 2 5 0 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75))) wh
ich suggests (APPROACH-TARGET)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (APPROACH-TARGET
GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 0 3.0 0 4 ?GRIPPING-FORCE2354 5 F1 F3 0 0)...
Suggestions for ADJUST-FINGERS-FOR-APPROACH are (PREPARE-FOR-APPROACH)...
Working on (ADJUST-FINGERS-FOR-APPROACH GRIPPER1 -9.5 3.5 0 2 3.75 2 3 4 2 5 2 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1

```
3
.75))) which suggests (PREPARE-FOR-APPROACH)...Attempting Suggestion Link-Up...Re-
trieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to
prove suggestion...Explanation found...Queueing (PR
EPARE-FOR-APPROACH GRIPPER1 -9.5 3.5 0 3.75 2 3 4 2 5 0 3 (GRIPPER1 SQUARE1
SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-AN-
GLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for ACHIEVE-FRICTIONAL-FORCE are (GRASP)...
Working on (ACHIEVE-FRICTIONAL-FORCE GRIPPER1 SQUARE2 5 0 5 ?GRIP-
PING-FORCE2032 5) which suggests (GRASP)...Attempting Suggestion Link-Up...Retrieving
Suggested Rule...Found Corresponding Antecedent...Successful...Working to
p-down to prove suggestion...Explanation found...Queueing (GRASP GRIPPER1 SQUARE2
?GRIPPING-FORCE2032 5 0 5)...
Suggestions for GRASP are NIL...
Working on (GRASP GRIPPER1 SQUARE2 ?GRIPPING-FORCE2032 5 0 5) which suggests
NIL...Enter Name For Cache Assignment-->Initial Non-tolerant
Generalizing An Explanation...
Cleaning Up Rules & Object Hierarchy...
;;; Loading source file "rules.lisp"
;;; Warning: Redefining function LIKELY-DEMON
Installing Learned Rule...
Asking System To Grasp An Object...

GRASP
  GRIPPER1 <---------- Gripper Name
  SQUARE2 <---------- Object
  5 <---------- Force For Gripping
  5 <---------- Force For Gripper Movements
  0 <---------- Numeric Starting Time
  ?E <---------- Numeric Ending Time

Plan Formulated...

EXECUTE-APPLY-FORCE
  GRIPPER1 <---------- Gripper Name
  -1.0 <---------- Delta X
  -5.0 <---------- Delta Y
  5 <---------- Force For Gripper Movements
  0 <---------- Numeric Starting Time
  1 <---------- Numeric Ending Time
  GRIPPER-CLEAR-TRANSLATE
    GRIPPER1 <---------- Gripper Name
    -7.5 <---------- Current Gripper X-Coordinate
    8.5 <---------- Current Gripper Y-Coordinate
    -8.5 <---------- Target X-Coordinate
    3.5 <---------- Target Y-Coordinate
    (GRIPPER1 SQUARE1 SQUARE2)
    4 <---------- Gripper Beam Length
    2 <---------- Gripper Beam Width
```

```
    3 <---------- Gripper Finger Length
    2 <---------- Gripper Finger Width
    2 <---------- Current Gripper Finger Separation
    -90 <---------- Current Gripper Angle
  AT-COORDINATES
    -8.5 <---------- X-Coordinate
    3.5 <---------- Y-Coordinate
    GRIPPER1 <---------- Gripper Name
  ?TERM-VALUES9119422401 <-- Dump Of Gripper State After The Operation


EXECUTE-APPLY-TORQUE
  GRIPPER1 <---------- Gripper Name
  90.0 <---------- Delta Angle
  5 <---------- Force For Gripper Movements
  1 <---------- Numeric Starting Time
  2 <---------- Numeric Ending Time
  GRIPPER-CLEAR-ROTATE
    GRIPPER1 <---------- Gripper Name
    -90 <---------- Current Gripper Angle
    0.0 <---------- Target Gripper Angle
    (GRIPPER1 SQUARE1 SQUARE2)
    4 <---------- Gripper Beam Length
    2 <---------- Gripper Beam Width
    3 <---------- Gripper Finger Length
    2 <---------- Gripper Finger Width
    2 <---------- Current Gripper Finger Separation
    -8.5 <---------- Current Gripper X-Coordinate
    3.5 <---------- Current Gripper Y-Coordinate
  AT-ANGLE
    0.0 <---------- Angle
    GRIPPER1 <---------- Gripper Name
  ?TERM-VALUES109611252405 <-- Dump Of Gripper State After The Operation


EXECUTE-OPEN-FINGERS
  GRIPPER1 <---------- Gripper Name
  1.0 <---------- Delta Width
  5 <---------- Force For Gripper Movements
  2 <---------- Numeric Starting Time
  3 <---------- Numeric Ending Time
  GRIPPER-CLEAR-FINGER-ADJUST
    GRIPPER1 <---------- Gripper Name
    2 <---------- Current Gripper Finger Separation
    3.0 <---------- Target Gripper Width
    (GRIPPER1 SQUARE1 SQUARE2)
    -8.5 <---------- Current Gripper X-Coordinate
    3.5 <---------- Current Gripper Y-Coordinate
    0.0 <---------- Current Gripper Angle
```

```
    2 <---------- Current Gripper Finger Separation
    2 <---------- Gripper Beam Width
    4 <---------- Gripper Beam Length
    2 <---------- Gripper Finger Width
    3 <---------- Gripper Finger Length
  AT-WIDTH
    3.0 <---------- Width
    GRIPPER1 <---------- Gripper Name
  ?TERMINATION-VALUES125112852409 <-- Dump Of Gripper State After The Operation


EXECUTE-APPLY-FORCE
  GRIPPER1 <---------- Gripper Name
  13.0 <---------- Delta X
  0.0 <---------- Delta Y
  5 <---------- Force For Gripper Movements
  3 <---------- Numeric Starting Time
  4 <---------- Numeric Ending Time
  GRIPPER-APPROACH-SEQUENCE
    GRIPPER1 <---------- Gripper Name
    13.0 <---------- Delta X
    0.0 <---------- Delta Y
    SQUARE2 <---------- Object
    F1 <---------- One Face Of The Object
    F3 <---------- Another Face Of The Object
    4.5 <---------- Target X-Coordinate
    3.5 <---------- Target Y-Coordinate
    0.0 <---------- Target Gripper Angle
    3.0 <---------- Target Gripper Width
    -8.5 <---------- Current Gripper X-Coordinate
    3.5 <---------- Current Gripper Y-Coordinate
    #<Structure POLY 10754DC3> <-- Polygon Representation Of The Object
    1.5 <---------- Nearest Gripper X-Position Prior To Object Contact
    3.5 <---------- Nearest Gripper Y-Position Prior To Object Contact
    (GRIPPER1 SQUARE1 SQUARE2)
    4 <---------- Gripper Beam Length
    2 <---------- Gripper Beam Width
    3 <---------- Gripper Finger Length
    2 <---------- Gripper Finger Width
    0.0 <---------- Current Gripper Angle
  AT-COORDINATES
    4.5 <---------- X-Coordinate
    3.5 <---------- Y-Coordinate
    GRIPPER1 <---------- Gripper Name
  ?TERMINATION-VALUES154515702420 <-- Dump Of Gripper State After The Operation


EXECUTE-CLOSE-FINGERS
  GRIPPER1 <---------- Gripper Name
```

```
  0.0 <---------- Delta Width
  ?FORCE19052422 <---- Force For Gripper Movements
  4 <---------- Numeric Starting Time
  5 <---------- Numeric Ending Time
  GRIPPING-SEQUENCE
    GRIPPER1 <---------- Gripper Name
    SQUARE2 <---------- Object
    F1 <---------- One Face Of The Object
    F3 <---------- Another Face Of The Object
    ?FORCE19052422 <-- Motor Force Applied
    3.0 <---------- Current Gripper Finger Separation
    3.0 <---------- Target Gripper Width
  GRIPPER-GRIP-PRESSURE
    GRIPPER1 <---------- Gripper Name
    ?FORCE19052422 <-- Object
  ?TERMINATION-VALUES187818892425 <-- Dump Of Gripper State After The Operation

Executing Plan...
Plan Failed!
  Prior Value Dump:
                (GRIPPER-X GRIPPER1 -8.5)
                (GRIPPER-Y GRIPPER1 3.5)
                (GRIPPER-ANGLE GRIPPER1 0.0)
                (GRIPPER-FINGER-SEPARATION GRIPPER1 3.0)
                (NOT (GRIPPER-X GRIPPER1 -7.5))
                (NOT (GRIPPER-Y GRIPPER1 8.5))
                (NOT (GRIPPER-ANGLE GRIPPER1 -90))
                (NOT (GRIPPER-FINGER-SEPARATION GRIPPER1 2))
  Dump At Failure:
                (GRIPPER-X GRIPPER1 1.5)
                (GRIPPER-Y GRIPPER1 3.5)
                (GRIPPER-ANGLE GRIPPER1 0)
                (GRIPPER-FINGER-SEPARATION GRIPPER1 3)
                (FORCE GRIPPER1 FINGER1 3 0.25 5 -1 0)
                (NOT (GRIPPER-X GRIPPER1 -7.5))
                (NOT (GRIPPER-Y GRIPPER1 8.5))
                (NOT (GRIPPER-ANGLE GRIPPER1 -90))
                (NOT (GRIPPER-FINGER-SEPARATION GRIPPER1 2))
  Expectation Goal:
    GRIPPER-APPROACH-SEQUENCE
      GRIPPER1 <---------- Gripper Name
      13.0 <---------- Delta X
      0.0 <---------- Delta Y
      SQUARE2 <---------- Object
      F1 <---------- One Face Of The Object
      F3 <---------- Another Face Of The Object
      4.5 <---------- Target X-Coordinate
      3.5 <---------- Target Y-Coordinate
      0.0 <---------- Target Gripper Angle
```

```
      3.0 <---------- Target Gripper Width
      -8.5 <---------- Current Gripper X-Coordinate
      3.5 <---------- Current Gripper Y-Coordinate
      #<Structure POLY 10754DC3> <-- Polygon Representation Of The Object
      1.5 <---------- Nearest Gripper X-Position Prior To Object Contact
      3.5 <---------- Nearest Gripper Y-Position Prior To Object Contact
      (GRIPPER1 SQUARE1 SQUARE2)
      4 <---------- Gripper Beam Length
      2 <---------- Gripper Beam Width
      3 <---------- Gripper Finger Length
      2 <---------- Gripper Finger Width
      0.0 <---------- Current Gripper Angle
    Expectation Expl:
     #<Structure EXPLANATION 10799D63>
    Expected Dump:
                    (GRIPPER-X GRIPPER1 1.5)
                    (GRIPPER-Y GRIPPER1 3.5)
                    (GRIPPER-ANGLE GRIPPER1 0)
                    (GRIPPER-FINGER-SEPARATION GRIPPER1 3)
                    (NOT (GRIPPER-X GRIPPER1 -7.5))
                    (NOT (GRIPPER-Y GRIPPER1 8.5))
                    (NOT (GRIPPER-ANGLE GRIPPER1 -90))
                    (NOT (GRIPPER-FINGER-SEPARATION GRIPPER1 2))
Explaining Failure(s)...
Evaluating SQUARE1's OBJECT-POSITION-APPROXIMATION With Respect To A Contact
At (3.0 5.25).
  - Contact Deviates From Approximation By 8.400148
Evaluating SQUARE1's OBJECT-SHAPE-APPROXIMATION With Respect To A Contact At
(3.0 5.25).
  - Contact Deviates From Approximation By 6.75
Evaluating SQUARE1's OBJECT-ORIENTATION-APPROXIMATION With Respect To A
Contact At (3.0 5.25).
  - Contact Not Explainable By Deviation In This Approximation
Evaluating SQUARE2's OBJECT-POSITION-APPROXIMATION With Respect To A Contact
At (3.0 5.25).
  - Contact Deviates From Approximation By 0.25
Evaluating SQUARE2's OBJECT-SHAPE-APPROXIMATION With Respect To A Contact At
(3.0 5.25).
  - Contact Deviates From Approximation By 0.25
Evaluating SQUARE2's OBJECT-ORIENTATION-APPROXIMATION With Respect To A
Contact At (3.0 5.25).
  - Contact Not Explainable By Deviation In This Approximation
4 Possible Failure(s) Have Been Identified.
The following is a list of possible root causes for the failure.
Failure of OBJECT-POSITION-APPROXIMATION with regard to SQUARE1
Failure of OBJECT-SHAPE-APPROXIMATION with regard to SQUARE1
Failure of OBJECT-POSITION-APPROXIMATION with regard to SQUARE2
Failure of OBJECT-SHAPE-APPROXIMATION with regard to SQUARE2
Reloading Rules & Object Hierarchy...
```

```
;;; Loading source file "rules.lisp"
;;; Warning: Redefining function LIKELY-DEMON
Observation Started...
Action Observed...

OBSERVED-APPLY-FORCE
  GRIPPER1 <---------- Gripper Name
  5 <---------- Motor Force Applied
  0 <---------- Numeric Starting Time
  1 <---------- Numeric Ending Time
  -7.5 <---------- Current Gripper X-Coordinate
  8.5 <---------- Current Gripper Y-Coordinate
  -9.5 <---------- Target X-Coordinate
  3.5 <---------- Target Y-Coordinate
  4 <---------- Gripper Beam Length
  2 <---------- Gripper Beam Width
  3 <---------- Gripper Finger Length
  2 <---------- Gripper Finger Width
  2 <---------- Current Gripper Finger Separation
  -90 <---------- Current Gripper Angle

Action Observed...

OBSERVED-APPLY-TORQUE
  GRIPPER1 <---------- Gripper Name
  5 <---------- Motor Force Applied
  1 <---------- Numeric Starting Time
  2 <---------- Numeric Ending Time
  -90 <---------- Current Gripper Angle
  0 <---------- Target Gripper Angle
  4 <---------- Gripper Beam Length
  2 <---------- Gripper Beam Width
  3 <---------- Gripper Finger Length
  2 <---------- Gripper Finger Width
  2 <---------- Current Gripper Finger Separation
  -9.5 <---------- Current Gripper X-Coordinate
  3.5 <---------- Current Gripper Y-Coordinate

Action Observed...

OBSERVED-OPEN-FINGERS
  GRIPPER1 <---------- Gripper Name
  5 <---------- Motor Force Applied
  2 <---------- Numeric Starting Time
  3 <---------- Numeric Ending Time
  2 <---------- Current Gripper Finger Separation
  3.75 <---------- Target Gripper Width
  -9.5 <---------- Current Gripper X-Coordinate
  3.5 <---------- Current Gripper Y-Coordinate
```

```
  0 <---------- Current Gripper Angle
  2 <---------- Current Gripper Finger Separation
  2 <---------- Gripper Beam Width
  4 <---------- Gripper Beam Length
  2 <---------- Gripper Finger Width
  3 <---------- Gripper Finger Length

Action Observed...

OBSERVED-APPLY-FORCE
  GRIPPER1 <---------- Gripper Name
  5 <---------- Motor Force Applied
  3 <---------- Numeric Starting Time
  4 <---------- Numeric Ending Time
  -0.5 <---------- Current Gripper X-Coordinate
  3.5 <---------- Current Gripper Y-Coordinate
  4.5 <---------- Target X-Coordinate
  3.5 <---------- Target Y-Coordinate
  4 <---------- Gripper Beam Length
  2 <---------- Gripper Beam Width
  3 <---------- Gripper Finger Length
  2 <---------- Gripper Finger Width
  3.75 <---------- Current Gripper Finger Separation
  0 <---------- Current Gripper Angle

Action Observed...

OBSERVED-CLOSE-FINGERS
  GRIPPER1 <---------- Gripper Name
  5 <---------- Motor Force Applied
  4 <---------- Numeric Starting Time
  5 <---------- Numeric Ending Time
  3.75 <---------- Current Gripper Finger Separation
  3 <---------- Target Gripper Width
  4.5 <---------- Current Gripper X-Coordinate
  3.5 <---------- Current Gripper Y-Coordinate
  0 <---------- Current Gripper Angle
  3.75 <---------- Current Gripper Finger Separation
  2 <---------- Gripper Beam Width
  4 <---------- Gripper Beam Length
  2 <---------- Gripper Finger Width
  3 <---------- Gripper Finger Length

Observation Ended...
Seeking To Understand Observations...
Seeking An Understanding Which Prevents Error With OBJECT-POSITION-APPROXIMATION
Of SQUARE2
;;; Loading source file "rules.lisp"
;;; Warning: Redefining function LIKELY-DEMON
```

Updating SQUARE2's OBJECT-POSITION-APPROXIMATION To Account For A Contact At (3.0 5.25).
That sequence of observations corresponds to cached understandings.
Please indicate whether you wish one to be retrieved.
Creating suggestion list...

Queue: (OBSERVED-APPLY-FORCE OBSERVED-APPLY-TORQUE OB-
SERVED-OPEN-FINGERS OBSERVED-APPLY-FORCE OBSERVED-CLOSE-FINGERS)

Suggestions for OBSERVED-APPLY-FORCE are (APPLY-FORCE)...
Working on (OBSERVED-APPLY-FORCE GRIPPER1 5 0 1 -7.5 8.5 -9.5 3.5 4 2 3 2 2 -90)
which suggests (APPLY-FORCE)...Attempting Suggestion Link-Up...Retrieving Suggested
Rule...Found Corresponding Antecedent...Successful...Found
Corresponding Antecedent...Successful...Working top-down to prove suggestion...Working
top-down to prove suggestion...Explanation found...Queueing (APPLY-FORCE GRIPPER1 -2.0
-5.0 5 0 1 (GRIPPER-CLEAR-TRANSLATE GRIPPER1 -7.5
8.5 -9.5 3.5 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 2 -90) (AT-COORDINATES -9.5
3.5 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIP-
PER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Suggestions for OBSERVED-APPLY-TORQUE are (APPLY-TORQUE)...
Working on (OBSERVED-APPLY-TORQUE GRIPPER1 5 1 2 -90 0 4 2 3 2 2 -9.5 3.5)
which suggests (APPLY-TORQUE)...Attempting Suggestion Link-Up...Retrieving Suggested
Rule...Found Corresponding Antecedent...Successful...Working top
-down to prove suggestion...Explanation found...Queueing (APPLY-TORQUE GRIPPER1 90.0
5 1 2 (GRIPPER-CLEAR-ROTATE GRIPPER1 -90 0 (GRIPPER1 SQUARE1 SQUARE2) 4 2
3 2 2 -9.5 3.5) (AT-ANGLE 0 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5)
(GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPA-
RATION GRIPPER1 2)))...
Suggestions for OBSERVED-OPEN-FINGERS are (OPEN-FINGERS)...
Working on (OBSERVED-OPEN-FINGERS GRIPPER1 5 2 3 2 3.75 -9.5 3.5 0 2 2 4 2 3)
which suggests (OPEN-FINGERS)...Attempting Suggestion Link-Up...Retrieving Suggested
Rule...Found Corresponding Antecedent...Successful...Working
top-down to prove suggestion...Explanation found...Queueing (OPEN-FINGERS GRIPPER1
1.75 5 2 3 (GRIPPER-CLEAR-FINGER-ADJUST GRIPPER1 2 3.75 (GRIPPER1 SQUARE1
SQUARE2) -9.5 3.5 0 2 2 4 2 3) (AT-WIDTH 3.75 GRIPPER1) ((GRIPPER-X
GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIP-
PER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for OBSERVED-APPLY-FORCE are (APPLY-FORCE)...
Working on (OBSERVED-APPLY-FORCE GRIPPER1 5 3 4 -9.5 3.5 4.5 3.5 4 2 3 2 3.75 0)
which suggests (APPLY-FORCE)...Attempting Suggestion Link-Up...Retrieving Suggested
Rule...Found Corresponding Antecedent...Successful...Found
Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation
found...Queueing (APPLY-FORCE GRIPPER1 14.0 0.0 5 3 4 (GRIPPER-APPROACH-SE-
QUENCE GRIPPER1 14.0 0.0 SQUARE2 F1 F3 4.5 3.5 0 3.75 -9.5
3.5 ?OB-POLY3281 ?GCX3282 ?GCY3283 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 0)
(AT-COORDINATES 4.5 3.5 GRIPPER1) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y
GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIP-
PER1 3.7
5)))...

Suggestions for OBSERVED-CLOSE-FINGERS are (CLOSE-FINGERS)...
Working on (OBSERVED-CLOSE-FINGERS GRIPPER1 5 4 5 3.75 3 4.5 3.5 0 3.75 2 4 2 3) which suggests (CLOSE-FINGERS)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Work
ing top-down to prove suggestion...Explanation found...Queueing (CLOSE-FINGERS GRIPPER1 -0.75 5 4 5 (GRIPPING-SEQUENCE GRIPPER1 SQUARE2 F1 F3 5 3.75 3) (GRIPPER-GRIP-PRESSURE GRIPPER1 5) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3)))...

Queue: (APPLY-FORCE APPLY-TORQUE OPEN-FINGERS APPLY-FORCE CLOSE-FINGERS)

Suggestions for APPLY-FORCE are (MOVE-GRIPPER-CLEAR APPROACH-TARGET-CLEAR)...
Working on (APPLY-FORCE GRIPPER1 -2.0 -5.0 5 0 1 (GRIPPER-CLEAR-TRANSLATE GRIPPER1 -7.5 8.5 -9.5 3.5 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 2 -90) (AT-COORDINATES -9.5 3.5 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))) which suggests (MOVE-GRIPPER-CLEAR APPROACH-TARGET-CLEAR)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecede
nt...Successful...Found Corresponding Antecedent...Unsuccessful...Working top-down to prove suggestion...Explanation found...Queueing (MOVE-GRIPPER-CLEAR GRIPPER1 -2.0 -5.0 -7.5 8.5 -90 2 -9.5 3.5 2 3 4 2 5 0 1 (GRIPPER1 SQU
ARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Unsuccessful...Suggestions for APPLY-TORQUE are (ROTATE-GRIPPER-CLEAR)...
Working on (APPLY-TORQUE GRIPPER1 90.0 5 1 2 (GRIPPER-CLEAR-ROTATE GRIPPER1 -90 0 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 2 -9.5 3.5) (AT-ANGLE 0 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPE
R1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))) which suggests (ROTATE-GRIPPER-CLEAR)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Found Corresponding Antecedent.
..Unsuccessful...Working top-down to prove suggestion...Explanation found...Queueing (ROTATE-GRIPPER-CLEAR GRIPPER1 90.0 -9.5 3.5 -90 2 0 2 3 4 2 5 1 2 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER
1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Suggestions for OPEN-FINGERS are (ADJUST-FINGERS)...
Working on (OPEN-FINGERS GRIPPER1 1.75 5 2 3 (GRIPPER-CLEAR-FINGER-ADJUST GRIPPER1 2 3.75 (GRIPPER1 SQUARE1 SQUARE2) -9.5 3.5 0 2 2 4 2 3) (AT-WIDTH 3.75 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER
-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75))) which suggests (ADJUST-FINGERS)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Found Corresponding A
ntecedent...Unsuccessful...Working top-down to prove suggestion...Explanation found...Queue-

ing (ADJUST-FINGERS GRIPPER1 1.75 5 2 3 (GRIPPER-CLEAR-FINGER-ADJUST GRIPPER1 2 3.75 (GRIPPER1 SQUARE1 SQUARE2) -9.5 3.5 0 2 2 4 2 3)
(AT-WIDTH 3.75 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for APPLY-FORCE are (MOVE-GRIPPER-CLEAR APPROACH-TARGET-CLEAR)...
Working on (APPLY-FORCE GRIPPER1 14.0 0.0 5 3 4 (GRIPPER-APPROACH-SEQUENCE GRIPPER1 14.0 0.0 SQUARE2 F1 F3 4.5 3.5 0 3.75 -9.5 3.5 ?OB-POLY3281 ?GCX3282 ?GCY3283 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 0) (AT-COORDINATES 4.5 3.5 GRIPPER1) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75))) which suggests (MOVE-GRIPPER-CLEAR APPROACH-TARGET-CLEAR)...Attempting Suggestion Link-Up...
Retrieving Suggested Rule...Found Corresponding Antecedent...Unsuccessful...Found Corresponding Antecedent...Unsuccessful...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Success
ful...Working top-down to prove suggestion...Explanation found...Queueing (APPROACH-TARGET-CLEAR GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 2 3 4 2 5 3 4 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)) F1 F3)...
Suggestions for CLOSE-FINGERS are (GRIP ADJUST-FINGERS)...
Working on (CLOSE-FINGERS GRIPPER1 -0.75 5 4 5 (GRIPPING-SEQUENCE GRIPPER1 SQUARE2 F1 F3 5 3.75 3) (GRIPPER-GRIP-PRESSURE GRIPPER1 5) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3))) which suggests (GRIP ADJUST-FINGERS)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation
found...Queueing (GRIP GRIPPER1 5 SQUARE2 F1 F3 4 5 3.75 3)...
Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Unsuccessful...Found Corresponding Antecedent...Unsuccessful...

Queue: (MOVE-GRIPPER-CLEAR ROTATE-GRIPPER-CLEAR ADJUST-FINGERS APPROACH-TARGET-CLEAR GRIP)

Suggestions for MOVE-GRIPPER-CLEAR are (MOVE-FOR-APPROACH)...
Working on (MOVE-GRIPPER-CLEAR GRIPPER1 -2.0 -5.0 -7.5 8.5 -90 2 -9.5 3.5 2 3 4 2 5 0 1 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION G
RIPPER1 2))) which suggests (MOVE-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing
(MOVE-FOR-APPROACH GRIPPER1 -7.5 8.5 -90 2 -9.5 3.5 2 3 4 2 5 0 1 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Suggestions for ROTATE-GRIPPER-CLEAR are (ROTATE-FOR-APPROACH)...
Working on (ROTATE-GRIPPER-CLEAR GRIPPER1 90.0 -9.5 3.5 -90 2 0 2 3 4 2 5 1 2 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1

3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))
which suggests (ROTATE-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (ROTATE-FO
R-APPROACH GRIPPER1 -9.5 3.5 -90 2 0 2 3 4 2 5 1 2 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Suggestions for ADJUST-FINGERS are (ADJUST-FINGERS-CLEAR)...
Working on (ADJUST-FINGERS GRIPPER1 1.75 5 2 3 (GRIPPER-CLEAR-FINGER-AD-
JUST GRIPPER1 2 3.75 (GRIPPER1 SQUARE1 SQUARE2) -9.5 3.5 0 2 2 4 2 3) (AT-WIDTH 3.75 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPP
ER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75))) which suggests (ADJUST-FINGERS-CLEAR)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Found Corresp
onding Antecedent...Unsuccessful...Working top-down to prove suggestion...Explanation found...Queueing (ADJUST-FINGERS-CLEAR GRIPPER1 1.75 ?GRIPPER-X4559 ?GRIP-
PER-Y4560 ?GRIPPER-ANGLE4561 2 3.75 2 3 4 2 5 2 3 (GRIPPER1 SQUAR
E1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIP-
PER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for APPROACH-TARGET-CLEAR are (APPROACH-TARGET)...
Working on (APPROACH-TARGET-CLEAR GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 2 3 4 2 5 3 4 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRI
PPER1 3.75)) F1 F3) which suggests (APPROACH-TARGET)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queu
eing (APPROACH-TARGET GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 0 3.0 0 4 ?GRIP-
PING-FORCE4601 5 F1 F3 0.25 0)...
Suggestions for GRIP are (ACHIEVE-FRICTIONAL-FORCE)...
Working on (GRIP GRIPPER1 5 SQUARE2 F1 F3 4 5 3.75 3) which suggests (ACHIEVE-FRICTIONAL-FORCE)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to
prove suggestion...Explanation found...Queueing (ACHIEVE-FRICTIONAL-FORCE GRIP-
PER1 SQUARE2 5 0 5 ?GRIPPING-FORCE4601 5)...

Queue: (MOVE-FOR-APPROACH ROTATE-FOR-APPROACH ADJUST-FINGERS-CLEAR APPROACH-TARGET ACHIEVE-FRICTIONAL-FORCE)

Suggestions for MOVE-FOR-APPROACH are (PREPARE-FOR-APPROACH)...
Working on (MOVE-FOR-APPROACH GRIPPER1 -7.5 8.5 -90 2 -9.5 3.5 2 3 4 2 5 0 1 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))
) which suggests (PREPARE-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (PREPARE
-FOR-APPROACH GRIPPER1 -9.5 3.5 0 3.75 2 3 4 2 5 0 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-AN-

GLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for ROTATE-FOR-APPROACH are (PREPARE-FOR-APPROACH)...
Working on (ROTATE-FOR-APPROACH GRIPPER1 -9.5 3.5 -90 2 0 2 3 4 2 5 1 2 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))) which suggests (PREPARE-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (PREPARE-FOR-AP
PROACH GRIPPER1 -9.5 3.5 0 3.75 2 3 4 2 5 0 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for ADJUST-FINGERS-CLEAR are (ADJUST-FINGERS-FOR-APPROACH)...
Working on (ADJUST-FINGERS-CLEAR GRIPPER1 1.75 -9.5 3.5 0 2 3.75 2 3 4 2 5 2 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.7 5))) which suggests (ADJUST-FINGERS-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queuein
g (ADJUST-FINGERS-FOR-APPROACH GRIPPER1 -9.5 3.5 0 2 3.75 2 3 4 2 5 2 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...

Suggestions for APPROACH-TARGET are (ACHIEVE-FRICTIONAL-FORCE)...
Working on (APPROACH-TARGET GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 0 3.0 0 4 ?GRIPPING-FORCE4601 5 F1 F3 0.25 0) which suggests (ACHIEVE-FRICTIONAL-FORCE)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (ACHIEVE-FRICTIONAL-FORCE GRIPPER1 SQUARE2 5 0 5 ?GRIPPING-FORCE4601 5)...
Suggestions for ACHIEVE-FRICTIONAL-FORCE are (GRASP)...
Working on (ACHIEVE-FRICTIONAL-FORCE GRIPPER1 SQUARE2 5 0 5 ?GRIPPING-FORCE4601 5) which suggests (GRASP)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working to
p-down to prove suggestion...Explanation found...Queueing (GRASP GRIPPER1 SQUARE2 ?GRIPPI'G-FORCE4601 5 0 5)...

Queue: (PREPARE-FOR-APPROACH ADJUST-FINGERS-FOR-APPROACH ACHIEVE-FRICTIONAL-FORCE GRASP)

Suggestions for PREPARE-FOR- APPROACH are (APPROACH-TARGET)...
Working on (PREPARE-FOR-APPROACH GRIPPER1 -9.5 3.5 0 3.75 2 3 4 2 5 0 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75))) wh ich suggests (APPROACH-TARGET)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (APPROACH-TARGET
GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 0 3.0 0 4 ?GRIPPING-FORCE4923 5 F1 F3 0.25 0)...

Suggestions for ADJUST-FINGERS-FOR-APPROACH are (PREPARE-FOR-APPROACH)...
Working on (ADJUST-FINGERS-FOR-APPROACH GRIPPER1 -9.5 3.5 0 2 3.75 2 3 4 2 5 2 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3
.75))) which suggests (PREPARE-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (PR
EPARE-FOR-APPROACH GRIPPER1 -9.5 3.5 0 3.75 2 3 4 2 5 0 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for ACHIEVE-FRICTIONAL-FORCE are (GRASP)...
Working on (ACHIEVE-FRICTIONAL-FORCE GRIPPER1 SQUARE2 5 0 5 ?GRIPPING-FORCE4601 5) which suggests (GRASP)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working to
p-down to prove suggestion...Explanation found...Queueing (GRASP GRIPPER1 SQUARE2 ?GRIPPING-FORCE4601 5 0 5)...
Suggestions for GRASP are NIL...
Working on (GRASP GRIPPER1 SQUARE2 ?GRIPPING-FORCE4601 5 0 5) which suggests NIL...Enter Name For Cache Assignment-->Position Approx Error
Seeking An Understanding Which Prevents Error With OBJECT-SHAPE-APPROXIMATION Of SQUARE2
;;; Loading source file "rules.lisp"
;;; Warning: Redefining function LIKELY-DEMON
Updating SQUARE2's OBJECT-SHAPE-APPROXIMATION To Account For A Contact At (3.0 5.25).
That sequence of observations corresponds to cached understandings.
Please indicate whether you wish one to be retrieved.
Creating suggestion list...

Queue: (OBSERVED-APPLY-FORCE OBSERVED-APPLY-TORQUE OBSERVED-OPEN-FINGERS OBSERVED-APPLY-FORCE OBSERVED-CLOSE-FINGERS)

Suggestions for OBSERVED-APPLY-FORCE are (APPLY-FORCE)...
Working on (OBSERVED-APPLY-FORCE GRIPPER1 5 0 1 -7.5 8.5 -9.5 3.5 4 2 3 2 2 -90) which suggests (APPLY-FORCE)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Found
Corresponding Antecedent...Successful...Working top-down to prove suggestion...Working top-down to prove suggestion...Explanation found...Queueing (APPLY-FORCE GRIPPER1 -2.0 -5.0 5 0 1 (GRIPPER-CLEAR-TRANSLATE GRIPPER1 -7.5
8.5 -9.5 3.5 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 2 -90) (AT-COORDINATES -9.5 3.5 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Suggestions for OBSERVED-APPLY-TORQUE are (APPLY-TORQUE)...
Working on (OBSERVED-APPLY-TORQUE GRIPPER1 5 1 2 -90 0 4 2 3 2 2 -9.5 3.5) which suggests (APPLY-TORQUE)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top
-down to prove suggestion...Explanation found...Queueing (APPLY-TORQUE GRIPPER1 90.0 5 1 2 (GRIPPER-CLEAR-ROTATE GRIPPER1 -90 0 (GRIPPER1 SQUARE1 SQUARE2) 4 2

3 2 2 -9.5 3.5) (AT-ANGLE 0 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5)
(GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPA-
RATION GRIPPER1 2)))...
Suggestions for OBSERVED-OPEN-FINGERS are (OPEN-FINGERS)...
Working on (OBSERVED-OPEN-FINGERS GRIPPER1 5 2 3 2 3.75 -9.5 3.5 0 2 2 4 2 3)
which suggests (OPEN-FINGERS)...Attempting Suggestion Link-Up...Retrieving Suggested
Rule...Found Corresponding Antecedent...Successful...Working
top-down to prove suggestion...Explanation found...Queueing (OPEN-FINGERS GRIPPER1
1.75 5 2 3 (GRIPPER-CLEAR-FINGER-ADJUST GRIPPER1 2 3.75 (GRIPPER1 SQUARE1
SQUARE2) -9.5 3.5 0 2 2 4 2 3) (AT-WIDTH 3.75 GRIPPER1) ((GRIPPER-X
GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIP-
PER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for OBSERVED-APPLY-FORCE are (APPLY-FORCE)...
Working on (OBSERVED-APPLY-FORCE GRIPPER1 5 3 4 -9.5 3.5 4.5 3.5 4 2 3 2 3.75 0)
which suggests (APPLY-FORCE)...Attempting Suggestion Link-Up...Retrieving Suggested
Rule...Found Corresponding Antecedent...Successful...Found
Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation
found...Queueing (APPLY-FORCE GRIPPER1 14.0 0.0 5 3 4 (GRIPPER-APPROACH-SE-
QUENCE GRIPPER1 14.0 0.0 SQUARE2 F1 F3 4.5 3.5 0 3.75 -9.5
3.5 ?OB-POLY5291 ?GCX5292 ?GCY5293 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 0)
(AT-COORDINATES 4.5 3.5 GRIPPER1) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y
GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIP-
PER1 3.7
5)))...
Suggestions for OBSERVED-CLOSE-FINGERS are (CLOSE-FINGERS)...
Working on (OBSERVED-CLOSE-FINGERS GRIPPER1 5 4 5 3.75 3 4.5 3.5 0 3.75 2 4 2 3)
which suggests (CLOSE-FINGERS)...Attempting Suggestion Link-Up...Retrieving Suggested
Rule...Found Corresponding Antecedent...Successful...Work
ing top-down to prove suggestion...Explanation found...Queueing (CLOSE-FINGERS GRIP-
PER1 -0.75 5 4 5 (GRIPPING-SEQUENCE GRIPPER1 SQUARE2 F1 F3 5 3.75 3) (GRIP-
PER-GRIP-PRESSURE GRIPPER1 5) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y
GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIP-
PER1 3)))...

Queue: (APPLY-FORCE APPLY-TORQUE OPEN-FINGERS APPLY-FORCE CLOSE-FIN-
GERS)

Suggestions for APPLY-FORCE are (MOVE-GRIPPER-CLEAR APPROACH-TAR-
GET-CLEAR)...
Working on (APPLY-FORCE GRIPPER1 -2.0 -5.0 5 0 1 (GRIPPER-CLEAR-TRANSLATE
GRIPPER1 -7.5 8.5 -9.5 3.5 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 2 -90) (AT-COOR-
DINATES -9.5 3.5 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1
3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1
2))) which suggests (MOVE-GRIPPER-CLEAR APPROACH-TARGET-CLEAR)...Attempting
Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecede
nt...Successful...Found Corresponding Antecedent...Unsuccessful...Working top-down to prove
suggestion...Explanation found...Queueing (MOVE-GRIPPER-CLEAR GRIPPER1 -2.0 -5.0
-7.5 8.5 -90 2 -9.5 3.5 2 3 4 2 5 0 1 (GRIPPER1 SQU

ARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Unsuccessful...Suggestions for APPLY-TORQUE are (ROTATE-GRIPPER-CLEAR)...
Working on (APPLY-TORQUE GRIPPER1 90.0 5 1 2 (GRIPPER-CLEAR-ROTATE GRIPPER1 -90 0 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 2 -9.5 3.5) (AT-ANGLE 0 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPE
R1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))) which suggests (ROTATE-GRIPPER-CLEAR)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Found Corresponding Antecedent.
..Unsuccessful...Working top-down to prove suggestion...Explanation found...Queueing (ROTATE-GRIPPER-CLEAR GRIPPER1 90.0 -9.5 3.5 -90 2 0 2 3 4 2 5 1 2 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER
1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Suggestions for OPEN-FINGERS are (ADJUST-FINGERS)...
Working on (OPEN-FINGERS GRIPPER1 1.75 5 2 3 (GRIPPER-CLEAR-FINGER-ADJUST GRIPPER1 2 3.75 (GRIPPER1 SQUARE1 SQUARE2) -9.5 3.5 0 2 2 4 2 3) (AT-WIDTH 3.75 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75))) which suggests (ADJUST-FINGERS)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Found Corresponding A
ntecedent...Unsuccessful...Working top-down to prove suggestion...Explanation found...Queueing (ADJUST-FINGERS GRIPPER1 1.75 5 2 3 (GRIPPER-CLEAR-FINGER-ADJUST GRIPPER1 2 3.75 (GRIPPER1 SQUARE1 SQUARE2) -9.5 3.5 0 2 2 4 2 3)
(AT-WIDTH 3.75 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for APPLY-FORCE are (MOVE-GRIPPER-CLEAR APPROACH-TARGET-CLEAR)...
Working on (APPLY-FORCE GRIPPER1 14.0 0.0 5 3 4 (GRIPPER-APPROACH-SEQUENCE GRIPPER1 14.0 0.0 SQUARE2 F1 F3 4.5 3.5 0 3.75 -9.5 3.5 ?OB-POLY5291 ?GCX5292 ?GCY5293 (GRIPPER1 SQUARE1 SQUARE2) 4 2 3 2 0) (AT-COORDINATES 4.5 3.5 GRIPPER1) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75))) which suggests (MOVE-GRIPPER-CLEAR APPROACH-TARGET-CLEAR)...Attempting Suggestion Link-Up...
Retrieving Suggested Rule...Found Corresponding Antecedent...Unsuccessful...Found Corresponding Antecedent...Unsuccessful...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Success
ful...Working top-down to prove suggestion...Explanation found...Queueing (APPROACH-TARGET-CLEAR GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 2 3 4 2 5 3 4 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)) F1 F3)...
Suggestions for CLOSE-FINGERS are (GRIP ADJUST-FINGERS)...
Working on (CLOSE-FINGERS GRIPPER1 -0.75 5 4 5 (GRIPPING-SEQUENCE GRIPPER1 SQUARE2 F1 F3 5 3.75 3) (GRIPPER-GRIP-PRESSURE GRIPPER1 5) ((GRIPPER-X GRIP-

PER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FIN GER-SEPARATION GRIPPER1 3))) which suggests (GRIP ADJUST-FINGERS)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (GRIP GRIPPER1 5 SQUARE2 F1 F3 4 5 3.75 3)...
Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Unsuccessful...Found Corresponding Antecedent...Unsuccessful...

Queue: (MOVE-GRIPPER-CLEAR ROTATE-GRIPPER-CLEAR ADJUST-FINGERS APPROACH-TARGET-CLEAR GRIP)

Suggestions for MOVE-GRIPPER-CLEAR are (MOVE-FOR-APPROACH)...
Working on (MOVE-GRIPPER-CLEAR GRIPPER1 -2.0 -5.0 -7.5 8.5 -90 2 -9.5 3.5 2 3 4 2 5 0 1 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION G RIPPER1 2))) which suggests (MOVE-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (MOVE-FOR-APPROACH GRIPPER1 -7.5 8.5 -90 2 -9.5 3.5 2 3 4 2 5 0 1 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Suggestions for ROTATE-GRIPPER-CLEAR are (ROTATE-FOR-APPROACH)...
Working on (ROTATE-GRIPPER-CLEAR GRIPPER1 90.0 -9.5 3.5 -90 2 0 2 3 4 2 5 1 2 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))) which suggests (ROTATE-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (ROTATE-FO R-APPROACH GRIPPER1 -9.5 3.5 -90 2 0 2 3 4 2 5 1 2 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2)))...
Suggestions for ADJUST-FINGERS are (ADJUST-FINGERS-CLEAR)...
Working on (ADJUST-FINGERS GRIPPER1 1.75 5 2 3 (GRIPPER-CLEAR-FINGER-ADJUST GRIPPER1 2 3.75 (GRIPPER1 SQUARE1 SQUARE2) -9.5 3.5 0 2 2 4 2 3) (AT-WIDTH 3.75 GRIPPER1) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPP ER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75))) which suggests (ADJUST-FINGERS-CLEAR)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Found Corresp onding Antecedent...Unsuccessful...Working top-down to prove suggestion...Explanation found...Queueing (ADJUST-FINGERS-CLEAR GRIPPER1 1.75 ?GRIPPER-X6559 ?GRIPPER-Y6570 ?GRIPPER-ANGLE6571 2 3.75 2 3 4 2 5 2 3 (GRIPPER1 SQUAR E1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for APPROACH-TARGET-CLEAR are (APPROACH-TARGET)...
Working on (APPROACH-TARGET-CLEAR GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 2 3 4 2 5 3 4 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 4.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRI PPER1 3.75)) F1 F3) which suggests (APPROACH-TARGET)...Attempting Suggestion

Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queu
eing (APPROACH-TARGET GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 0 3.0 0 4 ?GRIP-PING-FORCE6611 5 F1 F3 0 0)...
Suggestions for GRIP are (ACHIEVE-FRICTIONAL-FORCE)...
Working on (GRIP GRIPPER1 5 SQUARE2 F1 F3 4 5 3.75 3) which suggests (ACHIEVE-FRICTIONAL-FORCE)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (ACHIEVE-FRICTIONAL-FORCE GRIP-PER1 SQUARE2 5 0 5 ?GRIPPING-FORCE6611 5)...

Queue: (MOVE-FOR-APPROACH ROTATE-FOR-APPROACH ADJUST-FINGERS-CLEAR APPROACH-TARGET ACHIEVE-FRICTIONAL-FORCE)

Suggestions for MOVE-FOR-APPROACH are (PREPARE-FOR-APPROACH)...
Working on (MOVE-FOR-APPROACH GRIPPER1 -7.5 8.5 -90 2 -9.5 3.5 2 3 4 2 5 0 1 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 -90) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))) which suggests (PREPARE-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queueing (PREPARE
-FOR-APPROACH GRIPPER1 -9.5 3.5 0 3.75 2 3 4 2 5 0 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-AN-GLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for ROTATE-FOR-APPROACH are (PREPARE-FOR-APPROACH)...
Working on (ROTATE-FOR-APPROACH GRIPPER1 -9.5 3.5 -90 2 0 2 3 4 2 5 1 2 (GRIP-PER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 2))) which suggests (PREPARE-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Sug-gested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove sug-gestion...Explanation found...Queueing (PREPARE-FOR-AP
PROACH GRIPPER1 -9.5 3.5 0 3.75 2 3 4 2 5 0 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for ADJUST-FINGERS-CLEAR are (ADJUST-FINGERS-FOR-APPROACH)...
Working on (ADJUST-FINGERS-CLEAR GRIPPER1 1.75 -9.5 3.5 0 2 3.75 2 3 4 2 5 2 3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.7 5))) which suggests (ADJUST-FINGERS-FOR-APPROACH)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation found...Queuein
g (ADJUST-FINGERS-FOR-APPROACH GRIPPER1 -9.5 3.5 0 2 3.75 2 3 4 2 5 2 3 (GRIP-PER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...

Suggestions for APPROACH-TARGET are (ACHIEVE-FRICTIONAL-FORCE)...
Working on (APPROACH-TARGET GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 0 3.0 0 4 ?GRIPPING-FORCE6611 5 F1 F3 0 0) which suggests (ACHIEVE-FRIC-

TIONAL-FORCE)...Attempting Suggestion Link-Up...Retrieving Suggested Rule...Found Co
rresponding Antecedent...Successful...Working top-down to prove suggestion...Explanation
found...Queueing (ACHIEVE-FRICTIONAL-FORCE GRIPPER1 SQUARE2 5 0 5 ?GRIP-
PING-FORCE6611 5)...
Suggestions for ACHIEVE-FRICTIONAL-FORCE are (GRASP)...
Working on (ACHIEVE-FRICTIONAL-FORCE GRIPPER1 SQUARE2 5 0 5 ?GRIP-
PING-FORCE6611 5) which suggests (GRASP)...Attempting Suggestion Link-Up...Retrieving
Suggested Rule...Found Corresponding Antecedent...Successful...Working to
p-down to prove suggestion...Explanation found...Queueing (GRASP GRIPPER1 SQUARE2
?GRIPPING-FORCE6611 5 0 5)...

Queue: (PREPARE-FOR-APPROACH ADJUST-FINGERS-FOR-APPROACH
ACHIEVE-FRICTIONAL-FORCE GRASP)

Suggestions for PREPARE-FOR-APPROACH are (APPROACH-TARGET)...
Working on (PREPARE-FOR-APPROACH GRIPPER1 -9.5 3.5 0 3.75 2 3 4 2 5 0 3 (GRIP-
PER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5)
(GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75))) wh
ich suggests (APPROACH-TARGET)...Attempting Suggestion Link-Up...Retrieving Suggested
Rule...Found Corresponding Antecedent...Successful...Working top-down to prove sugges-
tion...Explanation found...Queueing (APPROACH-TARGET
GRIPPER1 SQUARE2 -9.5 3.5 0 3.75 4.5 3.5 0 3.0 0 4 ?GRIPPING-FORCE6933 5 F1 F3 0
0)...
Suggestions for ADJUST-FINGERS-FOR-APPROACH are (PREPARE-FOR-APPROACH)...
Working on (ADJUST-FINGERS-FOR-APPROACH GRIPPER1 -9.5 3.5 0 2 3.75 2 3 4 2 5 2
3 (GRIPPER1 SQUARE1 SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIP-
PER1 3.5) (GRIPPER-ANGLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1
3
.75))) which suggests (PREPARE-FOR-APPROACH)...Attempting Suggestion Link-Up...Re-
trieving Suggested Rule...Found Corresponding Antecedent...Successful...Working top-down to
prove suggestion...Explanation found...Queueing (PR
EPARE-FOR-APPROACH GRIPPER1 -9.5 3.5 0 3.75 2 3 4 2 5 0 3 (GRIPPER1 SQUARE1
SQUARE2) ((GRIPPER-X GRIPPER1 -9.5) (GRIPPER-Y GRIPPER1 3.5) (GRIPPER-AN-
GLE GRIPPER1 0) (GRIPPER-FINGER-SEPARATION GRIPPER1 3.75)))...
Suggestions for ACHIEVE-FRICTIONAL-FORCE are (GRASP)...
Working on (ACHIEVE-FRICTIONAL-FORCE GRIPPER1 SQUARE2 5 0 5 ?GRIP-
PING-FORCE6611 5) which suggests (GRASP)...Attempting Suggestion Link-Up...Retrieving
Suggested Rule...Found Corresponding Antecedent...Successful...Working to
p-down to prove suggestion...Explanation found...Queueing (GRASP GRIPPER1 SQUARE2
?GRIPPING-FORCE6611 5 0 5)...
Suggestions for GRASP are NIL...
Working on (GRASP GRIPPER1 SQUARE2 ?GRIPPING-FORCE6611 5 0 5) which suggests
NIL...Enter Name For Cache Assignment-->Shape Error
Generalizing An Explanation...
Generalizing An Explanation...
Cleaning Up Rules & Object Hierarchy...
;;; Loading source file "rules.lisp"
;;; Warning: Redefining function LIKELY-DEMON

```
Updating SQUARE2's OBJECT-SHAPE-APPROXIMATION To Account For A Contact At
(3.0 5.25).
Installing Learned Rule...
Asking System To Grasp An Object...

GRASP
  GRIPPER1 <---------- Gripper Name
  SQUARE2 <---------- Object
  5 <---------- Force For Gripping
  5 <---------- Force For Gripper Movements
  0 <---------- Numeric Starting Time
  ?E <---------- Numeric Ending Time

Plan Formulated...

EXECUTE-APPLY-FORCE
  GRIPPER1 <---------- Gripper Name
  -1.25 <---------- Delta X
  -5.0 <---------- Delta Y
  5 <---------- Force For Gripper Movements
  0 <---------- Numeric Starting Time
  1 <---------- Numeric Ending Time
  GRIPPER-CLEAR-TRANSLATE
    GRIPPER1 <---------- Gripper Name
    -7.5 <---------- Current Gripper X-Coordinate
    8.5 <---------- Current Gripper Y-Coordinate
    -8.75 <---------- Target X-Coordinate
    3.5 <---------- Target Y-Coordinate
    (GRIPPER1 SQUARE1 SQUARE2)
    4 <---------- Gripper Beam Length
    2 <---------- Gripper Beam Width
    3 <---------- Gripper Finger Length
    2 <---------- Gripper Finger Width
    2 <---------- Current Gripper Finger Separation
    -90 <---------- Current Gripper Angle
  AT-COORDINATES
    -8.75 <---------- X-Coordinate
    3.5 <---------- Y-Coordinate
    GRIPPER1 <---------- Gripper Name
  ?TERM-VALUES546955006980 <-- Dump Of Gripper State After The Operation


EXECUTE-APPLY-TORQUE
  GRIPPER1 <---------- Gripper Name
  90.0 <---------- Delta Angle
  5 <---------- Force For Gripper Movements
  1 <---------- Numeric Starting Time
  2 <---------- Numeric Ending Time
  GRIPPER-CLEAR-ROTATE
```

```
    GRIPPER1 <-------- Gripper Name
    -90 <------------- Current Gripper Angle
    0.0 <------------- Target Gripper Angle
    (GRIPPER1 SQUARE1 SQUARE2)
    4 <--------------- Gripper Beam Length
    2 <--------------- Gripper Beam Width
    3 <--------------- Gripper Finger Length
    2 <--------------- Gripper Finger Width
    2 <--------------- Current Gripper Finger Separation
    -8.75 <----------- Current Gripper X-Coordinate
    3.5 <------------- Current Gripper Y-Coordinate
  AT-ANGLE
    0.0 <------------- Angle
    GRIPPER1 <-------- Gripper Name
  ?TERM-VALUES565456836984 <-- Dump Of Gripper State After The Operation


EXECUTE-OPEN-FINGERS
  GRIPPER1 <---------- Gripper Name
  1.25 <-------------- Delta Width
  5 <----------------- Force For Gripper Movements
  2 <----------------- Numeric Starting Time
  3 <----------------- Numeric Ending Time
  GRIPPER-CLEAR-FINGER-ADJUST
    GRIPPER1 <-------- Gripper Name
    2 <--------------- Current Gripper Finger Separation
    3.25 <------------ Target Gripper Width
    (GRIPPER1 SQUARE1 SQUARE2)
    -8.75 <----------- Current Gripper X-Coordinate
    3.5 <------------- Current Gripper Y-Coordinate
    0.0 <------------- Current Gripper Angle
    2 <--------------- Current Gripper Finger Separation
    2 <--------------- Gripper Beam Width
    4 <--------------- Gripper Beam Length
    2 <--------------- Gripper Finger Width
    3 <--------------- Gripper Finger Length
  AT-WIDTH
    3.25 <------------ Width
    GRIPPER1 <-------- Gripper Name
  ?TERMINATION-VALUES580958436988 <-- Dump Of Gripper State After The Operation


EXECUTE-APPLY-FORCE
  GRIPPER1 <---------- Gripper Name
  13.25 <------------- Delta X
  0.0 <--------------- Delta Y
  5 <----------------- Force For Gripper Movements
  3 <----------------- Numeric Starting Time
  4 <----------------- Numeric Ending Time
```

```
  GRIPPER-APPROACH-SEQUENCE
    GRIPPER1 <--------- Gripper Name
    13.25 <------------ Delta X
    0.0 <-------------- Delta Y
    SQUARE2 <---------- Object
    F1 <--------------- One Face Of The Object
    F3 <--------------- Another Face Of The Object
    4.5 <-------------- Target X-Coordinate
    3.5 <-------------- Target Y-Coordinate
    0.0 <-------------- Target Gripper Angle
    3.25 <------------- Target Gripper Width
    -8.75 <------------ Current Gripper X-Coordinate
    3.5 <-------------- Current Gripper Y-Coordinate
    #<Structure POLY 10FBA2AB> <-- Polygon Representation Of The Object
    1.375 <------------ Nearest Gripper X-Position Prior To Object Contact
    3.5 <-------------- Nearest Gripper Y-Position Prior To Object Contact
    (GRIPPER1 SQUARE1 SQUARE2)
    4 <---------------- Gripper Beam Length
    2 <---------------- Gripper Beam Width
    3 <---------------- Gripper Finger Length
    2 <---------------- Gripper Finger Width
    0.0 <-------------- Current Gripper Angle
  AT-COORDINATES
    4.5 <-------------- X-Coordinate
    3.5 <-------------- Y-Coordinate
    GRIPPER1 <--------- Gripper Name
  ?TERMINATION-VALUES610361286999 <-- Dump Of Gripper State After The Operation

EXECUTE-CLOSE-FINGERS
  GRIPPER1 <----------- Gripper Name
  -0.25 <-------------- Delta Width
  ?FORCE64847001 <----- Force For Gripper Movements
  4 <------------------ Numeric Starting Time
  5 <------------------ Numeric Ending Time
  GRIPPING-SEQUENCE
    GRIPPER1 <--------- Gripper Name
    SQUARE2 <---------- Object
    F1 <--------------- One Face Of The Object
    F3 <--------------- Another Face Of The Object
    ?FORCE64847001 <-- Motor Force Applied
    3.25 <------------- Current Gripper Finger Separation
    3.0 <-------------- Target Gripper Width
  GRIPPER-GRIP-PRESSURE
    GRIPPER1 <--------- Gripper Name
    ?FORCE64847001 <-- Object
  ?TERMINATION-VALUES645764687004 <-- Dump Of Gripper State After The Operation

Executing Plan...
Plan Worked Successfully
```

REFERENCES

[Andreae84] P. M. Andreae, "Justified Generalization: Acquiring Procedures from Examples," Ph. D. Dissertation, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, January 1984. (Also appears as Technical Report 834, MIT AI Laboratory.)

[Bennett87] S. W. Bennett, "Approximation in Mathematical Domains," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987, pp. 239–241. (Also appears as Technical Report UILU–ENG–87–2238, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana–Champaign.)

[Braverman88] M. S. Braverman and S. J. Russell, "Boundaries of Operationality," *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI, June 1988, pp. 221–234.

[Brooks82] R. A. Brooks, "Symbolic Error Analysis and Robot Planning," Memo 685, MIT AI Lab, Cambridge, MA, September 1982.

[Brost88] R. C. Brost, "Automatic Grasp Planning in the Presence of Uncertainty," *The International Journal of Robotics Research 7*, 1 (February 1988), pp. 3–17.

[Chien87] S. A. Chien, "Simplifications in Temporal Persistence: An Approach to the Intractable Domain Theory Problem in Explanation–Based Learning," M.S. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, August 1987. (Also appears as UILU–ENG–87–2255, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana–Champaign.)

[Chien88] S. A. Chien, "Using and Refining Simplifications: Explanation–based Learning of Plans in Intractable Domains," Working Paper 85, AI Research Group, Coordinated Science Laboratory, University of Illinois, Urbana, IL, March 1988.

[DeJong86] G. F. DeJong and R. J. Mooney, "Explanation–Based Learning: An Alternative View," *Machine Learning 1*, 2 (April 1986), pp. 145–176. (Also appears as Technical Report UILU–ENG–86–2208, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana–Champaign.)

[Dietterich86] T. G. Dietterich, "Learning at the Knowledge Level," *Machine Learning 1*, 3 (1986), pp. 287–316.

[Dietterich88] T. G. Dietterich and J. S. Bennett, "Varieties of Operationality," 88–30–6, Oregon State University, June 6, 1988.

[Doyle86] R. J. Doyle, "Constructing and Refining Causal Explanations from an Inconsistent Domain Theory," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986, pp. 538–544.

[Ellman88] T. Ellman, "Explanation–Directed Search for Simplifying Assumptions," *Proceedings of the 1988 American Association for Artificial Intelligence Spring Symposium Series on Explanation–based Learning*, Stanford, CA, March 1988, pp. 95–99.

[Fikes72] R. E. Fikes, P. E. Hart and N. J. Nilsson, "Learning and Executing Generalized Robot Plans," *Artificial Intelligence 3*, 4 (1972), pp. 251–288.

[Gupta87] A. Gupta, "Explanation–Based Failure Recovery," *Proceedings of the National Conference on Artificial Intelligence*, Seattle, WA, July 1987, pp. 606–610.

[Hammond86] K. Hammond, "CHEF: A Model of Case–Based Planning," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986, pp. 267–271.

[Hirsh88] H. Hirsh, "Reasoning About Operationality for Explanation–Based Learning," *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI, June 1988, pp. 214–220.

[Holder88] L. B. Holder, "Discovering Substructure in Examples," M.S. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1988.

[Keller87a] R. M. Keller, "Defining Operationality for Explanation–Based Learning," *Proceedings of the National Conference on Artificial Intelligence*, Seattle, WA, July 1987, pp. 482–487.

[Keller87b] R. M. Keller, "The Role of Explicit Contextual Knowledge in Learning Concepts to Improve Performance," Ph.D. Dissertation, Department of Computer Science, Rutgers University, New Brunswick, NJ, January 1987. (Also appears as Machine Learning Technical Report 7, Laboratory for Computer Science Research, Rutgers University.)

[Keller87c] R. M. Keller, "Concept Learning in Context," *Proceedings of the Fourth International Workshop on Machine Learning*, University of California, Irvine, June 1987, pp. 91–102.

[Keller88] R. Keller, "Operationality and Generality in Explanation–Based Learning: Separate Dimensions or Opposite Endpoints?," *Proceedings of the 1988 American Association for Artificial Intelligence Spring Symposium Series on Explanation–based Learning*, Stanford, CA, March 1988, pp. 153–157.

[Mel88] B. W. Mel, "Building and Using Mental Models in a Sensory–Motor Domain: A Connectionist Approach," *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI, June 1988, pp. 207–213.

[Minton85] S. N. Minton, "Selectively Generalizing Plans for Problem-Solving," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985, pp. 596-599.

[Minton87] S. N. Minton and J. G. Carbonell, "Strategies for Learning Search Control Rules: An Explanation-based Approach," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987, pp. 228-235.

[Minton88] S. N. Minton, "Learning Effective Search Control Knowledge: An Explanation-Based Approach," CMU-CS-88-133, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, March 1988.

[Mitchell86] T. M. Mitchell, R. Keller and S. Kedar-Cabelli, "Explanation-Based Generalization: A Unifying View," *Machine Learning 1*, 1 (January 1986), pp. 47-80.

[Mooney86] R. J. Mooney and S. W. Bennett, "A Domain Independent Explanation-Based Generalizer," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986, pp. 551-555. (Also appears as Technical Report UILU-ENG-86-2216, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

[Mooney88] R. J. Mooney, "A General Explanation-Based Learning Mechanism and Its Application to Narrative Understanding," Ph.D. Dissertation, Department of Computer Science, University of Illinois, Urbana, IL, January 1988. (Also appears as UILU-ENG-87-2269, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

[Mostow83] D. J. Mostow, "Machine Transformation of Advice into a Heuristic Search Procedure,"in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (ed.), Tioga Publishing Company, Palo Alto, CA, 1983, pp. 367-404.

[Mostow87] J. Mostow and T. Fawcett, "Approximating Intractable Theories: A Problem Space Model," Machine Learning Technical Report 16, Department of Computer Science, Rutgers University, New Brunswick, NJ, December 1987.

[O'Rorke87] P. V. O'Rorke, "Explanation-Based Learning Via Constraint Posting and Propagation," Ph.D. Dissertation, Department of Computer Science, University of Illinois, Urbana, IL, January 1987. (Also appears as UILU-ENG-87-2239, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

[Prieditis87] A. E. Prieditis and J. Mostow, "PROLEARN: Towards a Prolog Interpreter that Learns," *Proceedings of the National Conference on Artificial Intelligence*, Seattle, WA, July 1987, pp. 494–498.

[Segre87] A. M. Segre, "Explanation–Based Learning of Generalized Robot Assembly Tasks," Ph.D. Dissertation, Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL, January 1987. (Also appears as UILU–ENG–87–2208, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana–Champaign.)

[Segre88] A. M. Segre, "Operationality and Real–World Plans," *Proceedings of the 1988 American Association for Artificial Intelligence Spring Symposium Series on Explanation–based Learning*, Stanford, CA, March 1988, pp. 158–163.

[Shavlik88] J. W. Shavlik, "Generalizing the Structure of Explanations in Explanation–Based Learning," Ph.D. Dissertation, Department of Computer Science, University of Illinois, Urbana, IL, January 1988. (Also appears as UILU–ENG–87–2276, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana–Champaign.)

[Sussman73] G. J. Sussman, "A Computational Model of Skill Acquisition," Technical Report 297, MIT AI Lab, Cambridge, MA, 1973.

[Whitehall87] B. L. Whitehall, "Substructure Discovery in Executed Action Sequences," M.S. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1987. (Also appears as Technical Report UILU–ENG–87–2256.)

[Zweben88] M. Zweben, "Improving Operationality with Approximate Heuristics," *Proceedings of the 1988 American Association for Artificial Intelligence Spring Symposium Series on Explanation–based Learning*, Stanford, CA, March 1988, pp. 100–106.